Qualitative modelling and simulation of physical systems for a diagnostic purpose

David Rozier
Department of Computer and Information Sciences
De Montfort University Milton Keynes

Dissertation submitted to De Montfort University
in partial fulfilment of the requirements for the degree of PhD

September 1998

Abstract

The goal of a fault-diagnosis system is to obtain an accurate diagnosis at a low cost. In order to reach this goal, many techniques have been used, *e.g.* qualitative methods and multiple-models. This research investigates a novel strategy for improving the balance accuracy versus cost of consistency-based fault-diagnosis systems.

This new strategy is organised around the notion of entities. These are physical entities, such as water pressure or temperature. The functioning of a physical system can involve numerous entities. Because these entities influence each other's behaviour, multiple-fault situations can occur, where several entities are affected by a fault. These situations are called complex multiple-fault situations.

The existing fault-diagnosis systems do not perform satisfactorily on complex multiple-fault situations. This is because the set of entities they investigate is fixed from the start of the diagnostic process. As a consequence, depending on the entities included in this set, existing systems either perform an inaccurate diagnosis, or reach an accurate diagnosis at an unnecessarily high cost. This thesis presents a fault-diagnosis strategy called MVDS (standing for Multiple Variable Diagnosis Strategy) designed specifically for performing the efficient diagnosis of complex multiple-fault situations. The underlying principle of MVDS is that it is not possible to know from the start of the diagnostic process which entities are affected. Thus, a diagnostic process with MVDS is undertaken with the investigation of an initial set of entities, and this set of investigated entities is continuously updated along the process, as intermediate results are obtained.

In order to illustrate clearly the functioning of MVDS, a fault-scenario using a small example from the air-conditioning domain is diagnosed and the process studied. The investigation of the performance of MVDS on more complex physical systems is undertaken on a larger case-study using a hot-water and heating system. In MVDS, it is possible to disable the adaptability of the set of investigated entities, so that it can be run with a fixed set. By doing so, the performance of the strategy in MVDS can be compared to the performance of traditional approaches which use a fixed set of investigated entities.

The study-case shows that MVDS reaches more accurate results than traditional approaches, and that this accuracy is obtained at a low cost, since unnecessary measurements of entities are avoided. Furthermore, the strategy produces a complete trace of the process that is close to common-sense reasoning. It is also a co-operative strategy where the operator can intervene.

Summary of the main research contributions:

- The issue of diagnosing complex multiple-fault situations is specifically addressed for the first time. The problem caused by this diagnosis task is defined, and a strategy is constructed in order to diagnose efficiently the complex multiple-fault situations. The strategy is implemented in MVDS and tested on an example and a case-study.
- Risk characteristics have been described. They allow to evaluate how prone to complex multiple-fault situations is a physical system.
- Hot-water and heating systems are offered as a new domain of research for consistency-based fault-diagnosis systems.
- The inclusion of co-operation into the fault-diagnosis process is a novel approach. Its potential advantages have been identified.

Acknowledgements

I would like to thank my director of studies, Dr. Kenneth Xia, for his firm and enthusiastic guidance of my research work and dissertation writing. This work has been supported by a research bursary from the School of Computing Sciences, De Montfort University.

I am grateful to Dr. Julie Dugdale, whose help with academic writing, time management and understanding of the PhD process has been invaluable. Her encouragement throughout these three years of research has been essential.

I would also like to acknowledge the help of Dr. Kamal Bechkoum, James Marshall and Dr. Jon Rowe. Their feed-back on the advance of my research and on the earlier versions of this dissertation has been painfully fair and rich.

Working in the Department of Computer and Information Sciences at Milton Keynes has been an enriching experience. I want to thank more particularly Pete Shepherd for his unconditional patience, generosity and professionalism in removing any potential technical obstacle to my progress.

I appreciated also the constant welcoming and supportive attitude of Dr. Gordon Clapworthy in response to the numerous solicitations for his advice as an esteemed Head of Research.

Finally but importantly, I thank all my family for their discrete although constant help, moral as well as financial. I am grateful to them for patiently supporting me during a total of eight years of studies at university, first at Grenoble and then at Milton Keynes.

Contents

List of figures

List of tables

Chapter 1

Artificial intelligence and fault-diagnosis systems

1.1 Introduction

1.1.1 Presentation of the research

The research described in this thesis is concerned with the fault-diagnosis of physical systems. Diagnosing a system in order to repair is an important task in many industries and domestic applications. When large and complex systems are of concern, computerised fault-diagnosis systems are used to perform the diagnosis. The result must be reliable and obtained in an economical manner. The more complex the faulty situation faced is, the more difficult it is to fulfil these two criteria.

Typically, physical systems' behaviour involve several entities, *e.g.* the water pressure, temperature, flow, content of air, hardness, in a water-distribution system. When several components are failing, and when several entities are affected, then the situation is called a complex multiple-fault situation. This research addresses the diagnosis of these specific situations, by setting-up a strategy, namely the Multiple-Variable Diagnosis Strategy (MVDS).

MVDS uses models of the correct behaviour within a consistency-based approach. In order to tackle the diagnosis of complex multiple-fault situations, MVDS exploits causal knowledge about the system's behaviour for guiding the management of multiple representations.

Diagnosing complex multiple-fault situations is an important issue because these situations can happen in industrial and domestic environments, and that there is not any satisfactory diagnosis strategy for them. As opposed to other strategies where inaccurate results are obtained or where unnecessary computation and measurements are undertaken, MVDS reaches accurate results while keeping the workload to a minimum.

A case-study has been developed in the domain of hot water and heating systems. It allows to run the PROLOG implementation of MVDS and evaluate its qualities and drawbacks. It demonstrates the strategy's ability to diagnose complex multiple-fault situations, and reveals a potential for co-operative problem solving.

The rest of Chapter 1 introduces the concepts linked to this research and situate this research in relation with these concepts. A detailed plan of this thesis is available in section 1.2, and a summary of the research aims and contributions is given in section 1.7.

1.1.2 Artificial intelligence

The field of artificial intelligence has been defined by Marvin Minsky as:

"The field of research concerned with making machines do things that people consider to require intelligence" [Minsky, 1987].

For many branches of artificial intelligence, this has been managed by conceiving computer systems which, to some extent, imitate human reasoning. This is the case for the domain of fault-diagnosis where major classes of systems use common-sense reasoning or qualitative reasoning.

The idea behind this work relates to the observation that human problem-solvers ignore pieces of knowledge that they believe are not of any use at the current state of the problem-solving process. This belief can be revisited at any stage of the process, so that so far ignored pieces of knowledge can then be used or kept unused until the end of the task. In all cases, unnecessary investigations about irrelevant pieces of knowledge are avoided.

This dynamic management of the available knowledge has rarely been studied and integrated in existing fault-diagnosis systems, at least not in a sufficient manner. Investigating a more satisfactory dynamic management of the available knowledge is the issue addressed in this work.

1.1.3 Faults and fault-diagnosis in physical systems

A physical system is a system which is not a computer system or a computer program. Fault-diagnosis of physical systems is a branch of Artificial Intelligence that is currently the subject of intensive research. The motivations behind this interest are fairly obvious, and stem from the need to diagnose a faulty system in order to repair it. Domains where computerised fault-diagnosis systems have been applied include the car industry [Price *et al.*, 1996; Struss *et al.*, 1996], the helicopter industry [Jammu *et al.*, 1998], telecommunication networks [Rozé, 1997], off-shore plants [Dressler *et al.*, 1993], the medical domain [Shortliffe, 1976; Miller *et al.*, 1982; Reggia *et al.*, 1984; Downing, 1991], and electro-mechanical [Milne *et al.*, 1996].

A widely accepted definition of a fault, encompassing most of the other definitions, is:

"A component is faulty if its correct behavior is inconsistent with the observations" [de Kleer & Williams, 1989]. This definition assumes that the component is observed individually, so that its current behaviour can be determined regardless of the behaviour of other components.

In other words, one is faced with a diagnostic problem as soon as the observations made on a physical system conflict with the way the system is meant to behave [Reiter, 1987].

The entire domain of research referred to as fault-diagnosis includes the task of fault-detection, which is the process of stating whether or not a system is faulty. Whereas the task of fault-diagnosis consists of determining *"why a correctly designed system is not functioning as it was intended"* [de Kleer *et al.*, 1992]. Therefore, it is assumed that the system is known to be faulty, *i.e.* that the fault-detection task is over. This task of fault-detection is thus not covered in this work.

Fault-diagnosis then includes two sub-tasks: fault-isolation and fault-identification [Coghill *et al.*, 1997]. Fault-isolation is the process of locating the faulty parts in the system. An example is to find

that the faulty part causing a car break-down is the carburettor. On the other hand, fault-identification is the process of identifying the nature of the faults, *i.e.* in what way the faulty components are failing. An example of fault-identification is to find that a blockage in a conduit is preventing any petrol to pass through, in a car's carburettor. Fault-isolation can be performed with no fault-identification, as in the example of stating that the carburettor is responsible. Fault-identification, however, can only be performed either at the same time as, or after, the fault-isolation, as in the previous example where finding the blockage cannot be found independently from its location in the carburettor.

In this work, the main function of the algorithm is fault-isolation, but the identification of the faults is immediately deduced from the fault-isolation results. Therefore, the system is a fault-diagnosis system, and not only a fault-isolation system.

1.1.4 Qualitative reasoning

Qualitative reasoning is reasoning about qualitative physics or naïve physics. This area of artificial intelligence was born in the mid-seventies with de Kleer's work on the NEWTON system, which performed qualitative simulations called Envisionments [de Kleer & Brown, 1984]. These were diagrams formed of sequences of states which the system may go through. Each sequence represented a possible behaviour of the physical system. It is still a highly active research field in artificial intelligence, and qualitative reasoning is used for solving numerous real-world problems [Lee & Ormsby, 1992; Travé-Massuyès and Milne, 1995]. In the same way that humans do not reason mainly by doing calculations on precise numbers, computer programs can use qualitative reasoning, and in this way perform tasks which could not be done with traditional numerical techniques. Indeed, these precise calculations can blur the important features of reasoning, *e.g.* distinguishing between behaviours in simulations [Parsons, 1993]. Avoiding this blur is the first aim of qualitative reasoning.

Another aim of qualitative reasoning is to create processes that can be easily understood by human operators. Indeed, qualitative reasoning allows output information to be formatted in a user-friendly manner without a complex interpretation process. This straightforward understanding of the output favours human-system co-operation and human verification.

The objects involved in qualitative reasoning are, for example, orders of magnitude [Raiman, 1991] or signs of numbers [de Kleer & Brown, 1984]. Manipulations on these objects are performed using specific algebras, for example interval algebras [Parsons, 1993] for order of magnitude manipulations.

Fault-diagnosis is an appropriate field of application for qualitative reasoning. First of all, human diagnosis experts do not perform numerical simulations of the system's behaviour. Rather, they try to gain a qualitative insight into the system's functioning by staying away from the numerical level of detail which prevents clear behavioural distinctions, indicating thereby the adequacy of qualitative reasoning for this task. Second, allowing co-operation and human-verification through an understandable common-sense process is also an important issue for fault-diagnosis, as their presence increases the trust of the human operator towards the results. This trust is necessary so that important decisions can be made on the basis of the diagnosis results. For these two reasons, this research uses qualitative reasoning in the fulfilment of the fault-diagnosis task.

1.2 Guide to the reader

In sections 1.3 and 1.4 of this chapter, it is shown that physical systems can develop complex multiple-fault situations, of which diagnosis have not been specifically addressed. The use of multiple representations for this task is suggested, in conjunction with the use of causal knowledge. A strategy for diagnosing these complex multiple-fault situations using multiple representations and causal knowledge has been implemented during this research work. This strategy is called MVDS, standing for Multiple Variable Diagnosis Strategy, and is introduced in section 1.5. In section 1.6, a water - distribution system is identified as a relevant physical system to apply MVDS on and evaluate the strategy's diagnostic performance. Finally, a listing of the aims of this research, and of the contributions it makes, can be found in section 1.7.

The underlying issues in model-based diagnosis, qualitative reasoning, multiple-fault situations, and causal information presented in the rest of this chapter form the basis of this research and are analysed in more detail in chapter 2. Existing diagnosis systems where dynamic management of multiple representations are used and where causal knowledge is utilised are described and studied.

In chapter 3, the domain of water distribution systems is explored and demonstrated to qualify for taking advantage from being diagnosed with MVDS. This is because it is shown to be subject to a class of multiple-fault situations referred to as complex multiple-fault situations. How an appropriate management of the set of the system's entities can address this problem is then studied. This investigation of the domain of application and of the requirements for the diagnosis of complex multiple-fault situations results in the explanation for MVDS' design.

The algorithm of MVDS is described in depth in chapter 4. The justifications for all intended features of MVDS are explained, together with the expected advantages and disadvantages.

In chapter 5, an instance of a water distribution system is described and taken as a case-study. Two fault-scenarios are constructed. They are diagnosed by using MVDS and some classic fault-diagnosis strategies, with three different sets of measurements. The result of each of the diagnostic processes is described, including an instance of MVDS' output to the operator.

The conclusions from these diagnostic tasks undertaken by MVDS are drawn in chapter 6. The successes and downfalls of these applications are analysed. In particular, the computational costs and the measurement costs associated with the different processes are compared.

The first part of chapter 7 summarises the work presented in this thesis. The contributions of the author to the field are listed and explained. In the second part, directions for further work are identified. Some of them are directly related to enhancing MVDS or improving management of the physical system's representation, and some are concerned with more general fault-diagnosis issues, which have been identified along this research.

1.3 Modelling and simulation for the diagnosis of physical systems

1.3.1 Model-based fault-diagnosis

Fault-diagnosis methods fall into two categories. One contains the rule-based methods, which use knowledge of faults, *e.g.* MYCIN [Shortliffe, 1976], INTERNIST [Miller *et al.*, 1982], and SYSTEM D [Reggia *et al.*, 1984]. The other contains the model-based methods, which use models of behaviours, *e.g.* SOPHIE [Brown *et al.*, 1982], DART [Genesereth, 1984], Davis' system [Davis, 1984], GDE [de Kleer & Williams, 1987], and DIAGNOSE [Reiter, 1987].

The rule-based methods have severe disadvantages, concerned with their explanation capabilities, the knowledge acquisition phase and their poor general diagnostic performance. These disadvantages, described in more detail in section 2.2.1, are the reasons why this research uses a model-based fault-diagnosis method.

When dealing with a model-based approach, the model of the system's correct behaviour which is used for the task is a major issue. Simulation of the model of the physical system provides its expected (or correct) behaviour, *i.e.* the behaviour that one would normally expect from the system. The observed behaviour is, on the other hand, the one currently noticed or measured. As explained in section 1.1.2, in a faulty physical system, the correct behaviour and the observed behaviour are inconsistent, and notification of this discrepancy is the fault-detection task. However, simulations of the expected behaviour of the physical system are also used for the fault-diagnosis task. Therefore, model-based fault-diagnosis is tightly linked with the modelling and simulating of the physical system. Indeed, comparisons between the expected and observed behaviours constitute the basis of the model-based fault-diagnosis task [Davis, 1983].

In the diagnostic process, the models are used for simulations. The type of model used and the nature of simulation performed drastically influence the result of the diagnostic process. Since it is tightly connected to the two sub-tasks of modelling and simulation, a successful fault-diagnosis depends on an appropriate fulfilment of these two sub-tasks.

Modelling and simulation of physical systems are non-trivial tasks. The two next sections describe two difficulties generally encountered about modelling and simulation, namely the diversity of valid models and the task-oriented aspect of simulation. The section 1.3.4 investigates then two other difficulties, concerned specifically with complex physical systems, namely their complexity, of course, and the incompleteness of the knowledge available. A solution to these difficulties is proposed in section 1.3.5.

1.3.2 Diversity of valid models

The first difficulty is that the number of models for a physical system is infinite. They may be more or less precise, more or less focused on certain features of the physical system. Still they are valid models for the same physical system, as they would be consistent with observations of the correct behaviour of the physical system. Given this fact, the modelling task consists of providing a valid and appropriate model, but there remains the question: What are the criteria that should be used, in order to assess if a model is appropriate or not ? The definition of these criteria relies on two pieces of information:

- What is the intended use of the model? The answer reveals the features of the physical system's behaviour that should be represented in the model, and also the level of detail [Dormoy & Raiman, 1988].
- What is the knowledge available for building the model? This knowledge is also called the "input information".

A model which takes into account all of the physical system's features could be seen as the perfectly appropriate model. However, building this model requires the input information to be complete. Unfortunately, this information can never be complete, as argued in section 1.3.4. This issue of what is required and what is provided is another important motivation for developing qualitative reasoning. This is due to the fact that building numerical models require an amount of knowledge which is often not available.

It has been seen that an appropriate model of the physical system must be selected for the fault-diagnosis task at hand. An efficient selection can be obtained through the use of multiple representations. This issue is discussed in section 1.3.5.

1.3.3 Simulating physical systems

Once a model of the physical system is built, the simulation task consists of identifying the expected behaviour of the physical system. It uses a model of the physical system and a set of input values. A simulation can be performed at different levels of detail. For example it might reveal only the important changes in the behaviour, or it might show any possibly minor variations. A simulation can also be focused on different features of the physical system. For instance, in an electrical circuit, it can be decided to simulate the values of the current intensity, and decide not to deal with the temperature of the components.

A simulation's output can take various forms. For example, a numerical output consists of a list of figures as the description of the behaviour, and can only be read by the simulation program designer. An example of a more readable result consists of text that an engineer, who has not taken part in building the simulation engine, can understand. This latter output format can be achieved by qualitative simulations, like de Kleer's Envisionments mentioned earlier [de Kleer, 1977]. Other possible formats are hybrid ones, between purely numerical and purely text formats.

As in the case of the definition of a model, the definition of a simulation engine is vague. Once again, a simulation engine cannot be designed without any given requirements. A simulation engine is task-oriented, and its construction interacts highly with the model that is available for the physical system and with the intended use of the simulations. This fact suggests the use of multiple models, as a manner to ensure that the simulations obtained out of the models always meet the current needs.

1.3.4 Complexity of the physical systems and incomplete knowledge

When dealing with physical systems, the terms 'variables' or 'parameters' are used in the literature with sometimes different meanings. This is why these terms are avoided in this dissertation as far as possible. Instead, the terms 'entities' and 'quantities' are used, with respect to the following examples:

Terminology:

An **entity** in a physical system is a physical phenomenon with no precision of location, *e.g.* the temperature of the water in a central-heating system.

A **quantity** is a couple (entity, location), *e.g.* the temperature of the water in the boiler in a central-heating system.

The two main characteristics of physical systems are now described. The first characteristic is the complexity of physical systems, since they are made of a large number of parts, interacting with each other using complicated paths of interaction. As a consequence, constructing a perfect model having exactly the same behaviour as the physical system is impossible to achieve. Therefore, building and using highly complex models of physical systems is a difficult task where perfection is unachievable. Thus, it is argued there is no reason for using unnecessarily complex models.

The second characteristic of physical systems is the difficulty in taking measurements. They are sometimes impossible to make and often expensive to obtain. There are various reasons for this. Firstly, the physical location of the point where a measurement is required can be difficult to reach in a large or highly complex physical system, for example for measurements in a hydraulic pipe behind a factory's wall. Secondly, measuring some quantities can be prevented by theoretical reasons, as for example the intensity of an electrical current, where plugging in a measuring device would modify the intensity to be measured. Thirdly, the measuring device might be complex and expensive to use, *e.g.* sensitive pressure or temperature sensors. Finally, it must be mentioned that a deficient machine might still be in use, and that the installation and use of the measuring device could mean stopping the use of this machine. This interruption means not only loss of money once again, but also social problems *e.g.* temporary unemployment. These are four reasons why typically only a limited set of measurements can be run on a physical system.

This limitation has two consequences. Firstly, highly complex models should be avoided, since many of the input values required for undertaking simulations cannot be measured. Therefore, a fault-diagnosis system should be able to reason with simple models. Secondly, a fault-diagnosis system should be able to reach a satisfactory result while keeping the number of measurements required as small as possible. These two consequences can actually be gathered in one, since the more complex

the models are, the larger the set of measurements required for their simulations is. Therefore, fault-diagnosis systems should "*use models as simple as possible and as sophisticated as necessary*" [Struss, 1992]. However, because physical systems can be highly complex, there lies a contradiction which motivates research into the issue of finding the simplest adequate model for a task to be performed on a physical system. One way of addressing this issue is by using multiple representations, or multiple models, as presented in the next section.

1.3.5 Multiple representations

In response to the difficulties described in sections 1.3.2 to 1.3.4, a fault-diagnosis system can use several models, or representations of the physical system at hand [Davis, 1982; Davis 1983; Struss, 1991], gathered in a set of models.

The fault-diagnosis task is started with a given model and some defined events can induce a change to the use of another model. In that way, no assumption needs to be made about the level of completeness or accuracy required for the fulfilment of the task. This is because this required level is to be identified during the process. The simpler model is typically used at the start of the task, and more complex models come into use when necessary, with respect to some criteria which are task-dependent.

The models in the set of models can, in theory, differ from each other by what components and variables they include and by the representation used for them. However, in practice, they typically differ in the latter manner, *e.g.* abstractions, simplifications, or approximations of each other [Struss, 1992; Dressler *et al.*, 1993]. It is argued that this is a restriction, because it should be examined which objects and variables need to be included in the model. This should be undertaken because it has been stated that these objects and variables are numerous, and therefore this is an obstacle to the required simplicity of the models used. For example, a common plumbing system includes multiple components, *e.g.* pipes, taps, vessels, heating devices and filters, and involves multiple entities, *e.g.* flow, pressure, temperature, content of air in the water, the content of rust in the water, the content of limescale in the water and the content of bacteria in the water. These different entities do not always need to be in the system's representation for every task, but should be included when evidence of their importance is gathered.

1.4 Multiple-fault situations and causality

It is commonly recognised that multiple-fault situations, *i.e.* situations where several faults are present, are less likely to occur than single-fault ones. Some fault-diagnosis systems even consider as impossible the occurrence of a multiple-fault situation: this is called the single-fault assumption (S.F.A.) [Genesereth, 1984]. A typical example is given in [Davis, 1983] where a diagnostic process is said to fail when "no single component is capable of explaining all the data". The basic reason for working under this single-fault assumption is that it is unlikely that two or more components would independently break down at the same time.

However components influence each other, and an important consequence of this influence is that the faulty behaviour of a component can induce another component's malfunction. In this case, because one fault causes another fault, and possibly resulting in a chain reaction, the breakdowns have not occurred at the same time, but eventually several components would be faulty together.

Obviously, this issue involves the factor of time. How many components are faulty at the moment of the diagnostic process? In order to stay with the assumption of a steady behaviour, it must be assumed that the time length of the diagnostic process is negligible compared to the time spans of evolution of the system's state. Hence, no further component breaks down while the diagnostic process takes place. Consider the time factor in more detail:

- if the diagnostic process always starts immediately after the initial breakdown occurs, then the assumption of the low likelihood of multiple faults is valid. This would be the case for successfully monitored systems, where a fault is detected as soon as it occurs and no other fault has the time to appear before repairs or compensatory actions are successful.

- in the usual case (non-monitored systems), the diagnostic process would start only when a fault is visible from the external world. That is to say when it concerns a functional and/or exogenous variable. If this breakdown is the result of a chain reaction, the diagnosis task faces a multiple-fault situation.

The occurrence of multiple-fault situations is therefore related to the interactions happening within the physical system. The paths followed by these interactions are usually called the paths of interaction or

causal paths [Davis, 1982; Davis, 1983]. In all cases, these paths are useful for improving the simulation of the system, but they can be even more important in the case of a multiple-fault situation. Indeed, because these interactions are directly responsible for the existence of several causally-linked faults, any knowledge about these paths is useful for the diagnosis strategy itself, and not only for the simulations. Knowledge of these paths is part of the kind of knowledge referred to as causal knowledge.

Diagnosing multiple-faults has been addressed in the literature, either because the possibility of these chains of faults was recognised [Trave-Massuyes & Milne, 1996], or just for the sake of addressing every possible situation, even highly improbable ones [de Kleer & Williams, 1987]. However, the next section shows that multiple-fault situations in physical systems can be intricate but no strategy has been established in order to perform an efficient fault-diagnosis for them.

1.5 The Multiple Variable Diagnosis Strategy (MVDS)

Multiple-fault situations have been shown in section 1.4 to be relevant problems for non-monitored physical systems. Furthermore, the causal paths supporting the interactions between components which can lead to multiple-fault situations are not constrained to stay within one single entity. For example, consider again the example of a plumbing system, the water limescale content is causally linked to the water air content, as water hardness tends to lead to high amounts of gas being formed. The direct consequence is that multiple-fault situations are likely to affect several entities.

Definition (Complex multiple-fault situation)

A situation where several faults are present, and where these faults involve several entities, is called a complex multiple-fault situation.

This definition is important because the diagnosis of complex multiple-fault situations is the central subject of this work. This has not been specifically addressed before. For this purpose, a diagnosis strategy, namely the Multiple-Variable Diagnosis Strategy (MVDS), is designed. The justifications for the various features that MVDS has been given are explained in detail in chapter 2, and the design itself of MVDS is described in chapter 3. Below is a brief presentation of MVDS.

MVDS is a strategy for the use of multiple representations for the diagnosis of complex multiple-fault situation. The strategy is a consistency-based method, following de Kleer and Brown's algorithm from GDE [de Kleer & Brown, 1987]. For the sake of remaining as close as possible to common-sense reasoning, it does not use Reiter's formalism [Reiter, 1987] described in section 2.2.3.2.

MVDS aims at providing an economical yet precise diagnosis of these situations. For this purpose, it makes use of causal knowledge as a guide for the efficient management of the multiple representations for the physical system. This causal knowledge is knowledge of the paths of influences between quantities or between entities, and can be empirical knowledge as well as deep-knowledge. The change of system's representation can intervene several times along an ongoing diagnostic process. These changes are called dynamic revisions, because the change performed depends on the current intermediate results obtained. However, the process does not start again from the beginning after a dynamic revision. Instead, the reasoning continues by using predictions made before and after the representation's change. Through this adaptation of the system's representation to the current state of the diagnostic process, it is aimed to use *"models as simple as possible and as sophisticated as necessary"* [Struss, 1992]

1.6 A domain of application: water-distribution systems

Domestic water-distribution systems are the systems which start at the main water inlet pipe of houses, and distribute water to one or several tanks, all the taps and appliances, *e.g.* washing machine and dish-washer. An instance of these domestic water systems is used in this work for applying MVDS for the following reasons.

First, these systems match several criteria which make them subject to multiple-fault situations:
- they are usually not monitored,
- they require some regular maintenance,
- the knowledge about some of the components involved is not complete,
- the conditions of use of some components are variables and uncontrolled.

Second, these systems involves many different entities, including the water flow, the water pressure, the water temperature, the water air content, the water rust content, the water limescale content and the water's bacterial content. Therefore, the multiple-fault situations developed are likely to be complex ones.

Third, the cost of making measurements on these water systems is high. Measuring any entity at any other point than at the end of the tap often means breaking open floors or ceilings and consuming many hours of labour. Avoiding investigation and thus measurement of some entities is therefore a valuable decrease in the diagnosis expense.

Finally, domestic water distribution systems are becoming more sophisticated, including water softeners and filters, pressure and flow controllers and dual water heating devices (gas or electrical combined with solar heating). The unavoidable consequence of this increased sophistication is a larger scope for faults, and the price of such complex systems requires appropriate diagnosis strategies for consumer satisfaction.

The reasons mentioned above therefore identify the domain of water distribution systems as suitable and relevant for the investigation of the performance of MVDS.

1.7 Aims and contributions of this research

1.7.1 Objectives

The high dependency of the diagnosis performance on the use of appropriate models and simulations is fundamental to this research. It results in the possibility of enhancing the efficiency of the diagnostic process through a more efficient use of the available modelling knowledge about the system.

Managing to enhance the efficiency of a system by improving its use of the available knowledge but keeping the core diagnostic reasoning as simple as possible helps to keep the whole process understandable by human-experts. Also the simple process is more likely to be able to produce useful explanations on how a certain result has been reached. This provides the diagnosis system with a valuable advantage for being applied in industrial circumstances.

The purpose of this research is to investigate how the different entities of a physical system should be managed in order to optimise their use towards a correct and efficient fault-diagnosis of complex multiple-fault situations. This concern of efficiently managing the entities is a novel research subject. The set of models for the entities to be managed is assumed to be given. Also the diagnosis of complex multiple-fault situations, defined in section 1.4, has not been addressed specifically before. Setting-up a strategy for managing a system's entities in order to diagnose efficiently complex multiple-fault situations is an important issue. This is because applying computerised fault-diagnosis systems to industrial problems means dealing with large and complex physical systems where the occurrence of these complex multiple-fault situations can be frequent, and where a reliable and economical diagnosis is required.

This research builds on the work on the use of multiple-representations by Davis and Struss, and on the use of a causal graph by Travé-Massuyès and Milne, and elaborates a strategy for the use of multiple representations and causal graph in the task of diagnosing complex multiple-fault situations.

The investigation for building this strategy MVDS is conducted through several steps:
- the specificity of diagnosing complex multiple-fault situations is studied,
- the important features which should be included in an efficient management of multiple representations are identified,
- MVDS is designed with respect to the two previous points,
- MVDS is implemented,
- a case-study is designed for the testing of MVDS,
- MVDS is used to diagnose several faulty situations on the case-study, including complex multiple-fault situations,
- the diagnostic processes performed by MVDS are examined and evaluated,
- conclusions are drawn about MVDS' advantages and weaknesses.

Along these steps, problems are met and solutions are formulated.

- Vague terms need to be avoided or defined precisely, thus the definitions for entities, quantities, complex multiple-fault situations, and dynamic revision.
- A case-study needs to be simple enough to illustrate clearly MVDS' algorithm, but complex enough to prove the efficiency of the strategy. As a consequence, two case-studies are used. A first one, simple, illustrates clearly how MVDS works, and a second one, more complex, proves its efficiency on non-trivial problems.

Eventually, it is aimed at evaluating the progress offered by MVDS for the task of diagnosing complex multiple-fault situations.

1.7.2 Summary of the research contributions

- Formal definitions of the terms 'entity' and 'quantity' are proposed, in the context of modelling and simulation of physical systems.
- Formal definitions of a 'complex multiple-fault situations' and of a 'diagnosis session' are proposed, in the context of the fault-diagnosis of physical systems.
- Risk criteria have been described. They allow to evaluate how prone to complex multiple-fault situations is a physical system.
- The issue of diagnosing complex multiple-fault situations has not been specifically addressed before.
- Containing models of the different entities is a new structure for the set of models.
- Using causal knowledge for guiding the management of the set of models is a novel approach.
- The criteria used for this management have not been used before.
- Hot-water and heating systems is a new domain of application for consistency-based fault-diagnosis systems.
- Allowing co-operation within the fault-diagnosis process and studying its advantages are novel issues.

Chapter 2

Background

2.1 Introduction

This chapter explores the fundamental issues related to this research and the related work.

In section 2.2, the task of fault-diagnosis itself is explored in depth. The task is defined and the various classes of fault-diagnosis systems are described. The advantages and disadvantages of each class are explained in detail. One of them, a sub-class of the model-based systems, contains consistency-based systems. The research concentrates on studying this latter sub-class of systems. The landmark papers on consistency-based systems and the various approaches are described in detail, together with their respective drawbacks.

In section 2.3, the issue of qualitative reasoning is examined. The motivations for using this kind of reasoning rather than more classic reasoning methods are explained. The various formalisms which have been set up for performing qualitative reasoning are then reviewed, analysed and compared.

The use of multiple representations and causal knowledge is investigated in section 2.4. The landmark works of Randall Davis and Peter Struss are described, and their weaknesses are identified.

2.2 Fault-diagnosis systems: the different classes

The task of fault diagnosis has been defined by David Poole as "the problem of trying to find what is wrong with some system, based on knowledge about the design/structure of the system, possible malfunctions that can occur in the system and observations (symptoms, evidence) made of the behaviour of the system" [Poole, 1989].

From this definition, it must be noticed that:
- the expected results of the task are not always fixed, as demonstrated by the use of the vague terms of "what is wrong",
- the type of knowledge used to accomplish the task is a major issue.

The definition of the task varies from one researcher to another, but the above two features are typically always present, and they are not completely independent. Consider the possible variations of the results. For example, one diagnosis can be the statement "Pipe number 2 is leaking" [Reggia *et al.*, 1984], and in another system, a diagnosis can be the assignment of a state, *e.g.* out of {normal, faulty}, to every component of the system [Reiter, 1987]. This difference is highly related to the nature of knowledge used in the reasoning. Thus the second feature of the definition, *i.e.* the nature of the available knowledge, is stressed again. It is actually the main factor to distinguish between classes of fault-diagnosis systems.

Figure 2.1 below represents the partition of fault-diagnosis systems into different classes. The partition of model-based systems into its two sub-classes is a discontinuous line because there are systems using hybrid consistency-based and abductive approaches [Console & Torasso, 1990; Struss & Dressler, 1989].

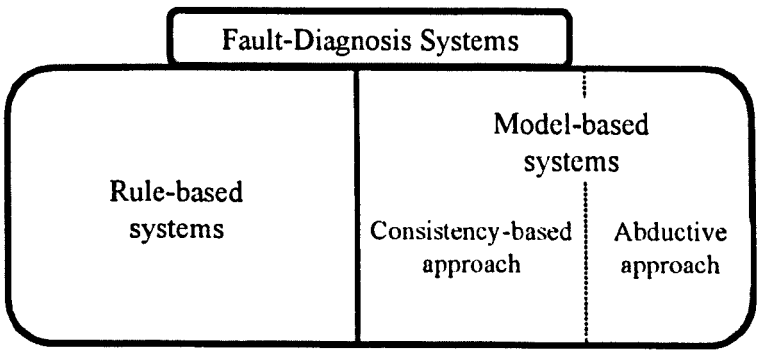


Figure 2.1 - Classes of fault-diagnosis systems

The two main classes of fault-diagnosis systems are rule-based systems and model-based systems. They are also called heuristic systems and deep-knowledge systems. These two classes are examined below, compared, and some major systems from each class are described.

2.2.1 Heuristic systems or rule-based systems

These fault-diagnosis systems use heuristic knowledge about the physical system, *i.e.* knowledge which has been acquired from past experience about the physical system's faults and associated symptoms. These heuristics are also called "rules of thumb". The diagnostic process reasons about analogies between the cases contained in the knowledge base and the symptoms noticed on the physical system to be diagnosed.

The diagnostic expert systems developed in the 70's are mainly heuristic systems. The knowledge representation formalisms they use vary from the simple *production rules* [Davis and King, 1977] to the more complex *frame* formalism [Minsky, 1975], and thus they differ in their inference strategies. The most influential expert systems constructed include the medical diagnostic expert systems MYCIN [Shortliffe, 1976], INTERNIST [Miller *et al.*, 1982], and SYSTEM D [Reggia *et al.*, 1984]. These systems have proven good diagnostic performance and speed, but only in limited domains [Davis, 1987; Coiera, 1992], and from the late 70's to early 80's severe limitations were shown [Torasso & Console, 1989].

The major drawback with heuristic systems is their inability to produce meaningful explanations as to how the problem was solved. They can only display the path of reasoning performed on the knowledge base, containing possible faults and their associated symptoms. This kind of explanation is insufficient in most cases. Consider the example where a car breaks down on a hot day and will not start again, and suppose that the driver manages to get his/her car started again by pouring cold water on the oil-pump, and that the reason he/she has done this diagnosis is because he/she already found himself in the same situation and solved it by pouring cold water on the oil-pump. He/she has used a heuristic rule, which is that if the car stops running on a hot day, then some water should be poured over the oil-pump. This explanation of the diagnosis, *i.e.* the statement of this rule, cannot be of any

use. What would be expected in this situation is an explanation of the mechanical reasons why pouring cold water on the oil-pump helps the engine to restart. However, producing such an explanation requires knowledge of a different nature than heuristic. The knowledge required is called deep-knowledge, and is knowledge about the structure and behaviour of the engine.

This inability to produce useful explanations is a major disadvantage of heuristic systems. In the case of a tutoring system, the production of deep and precise explanations is essential. In a more general frame, users need an appropriate explanation for the diagnosis system results in order to trust this advice and take decisions on the basis of it. It is considered that explanation is a primary task in problem-solving, rather than just an add-on facility [Wick, 1994]. As a consequence, a diagnosis system which cannot produce an appropriate explanation of its performance is highly limited in its use.

A second major problem with heuristic systems is a limitation in their diagnostic performance [Torasso & Console, 1989]. Indeed, they are efficient in diagnosing typical or simple situations, but are challenged in diagnosing unusual cases. This problem is caused by the fact that the knowledge about the possible faults is typically incomplete [Dague *et al.*, 1987], *i.e.* the set of rules does not cover all the possible fault cases. This is because heuristic knowledge captures the experience encountered to date in diagnosing the physical system [Davis, 1988], which is unlikely to include every possible fault. In order to diagnose the faulty situations which are not close enough to cases met in previous experiences, it is noticed again that knowledge about structure and behaviour of the physical system is needed.

Finally, the last important drawback is concerned with the purity of knowledge. In heuristic systems, the domain knowledge is mixed with control knowledge in order to increase control on the process. For example, the association between one symptom and a disease might be given more likelihood than it actually has, because the disease is very serious and no chance must be left to ignore the possibility. The impurity of the domain knowledge in heuristic systems makes the task of modifying and maintaining the knowledge base difficult. A diagnosis system is thus designed to be used on one particular application, and the re-usability of the system is diminished [Dague *et al.*, 1987]. It is important also to notice that, to some extent, a heuristic system is subjective, because the knowledge used is derived from the experience of a human expert. If the system was using knowledge about the structure and behaviour of the physical system, it would be more objective, *i.e.* the soundness and correctness of the domain knowledge could be more easily checked and validated by a number of experts.

These three important disadvantages of heuristic systems are major drawbacks to their performance. They are independent from the diagnosis system itself, because they are inherent in the use of heuristic knowledge. Thus all rule-based systems have these disadvantages. Since the late 70's and early 80's, these severe limitations of heuristic systems were noticed, and systems using a different kind of knowledge began to be built. They use knowledge about the structure and behaviour of the physical system, called deep-knowledge.

2.2.2 Systems using deep knowledge, or model-based systems

Model-based systems perform the diagnosis task using deep knowledge [Davis *et al.*, 1982; Genesereth, 1984]. It can be defined in a common-sense manner as a description of how the system is made and how it functions. The first important paper which founded this model-based paradigm dates back to 1982 [Davis *et al.*, 1982], in which Davis investigates the use of structural and behavioural descriptions for diagnosing digital electronics. This paper is addressed in detail in section 2.4.1.

The structure of a physical system is the description of its components and how they are linked. In every fault-diagnosis system, the respective notations used for describing the structure vary.

Regarding the behavioural description, the available knowledge influences what behavioural description will be given, and thus what diagnostic method will be used. In extreme cases, only some knowledge about the normal behaviour of the components is available, or there is also extensive knowledge about the possible faulty behaviour [Price, 1996; Price and Taylor, 1997]. In the first case, the method typically used follows consistency-based approach, and in the second case, abductive diagnosis is performed.

Within a consistency-based approach, a diagnosis is the smallest set of components so that the assumption that all these components are faulty and all the other ones are behaving correctly is consistent with the system description and the observations [Reiter, 1987].

On the other hand, an abductive diagnosis is a minimal set of abnormality assumptions which covers, *i.e.* implies, the observations [Console & Torasso, 1990; Paul, 1993].

The two approaches have been proven to be powerful [Poole, 1989], and the nature of the knowledge available is a guide towards what method is appropriate.

The disadvantage of consistency-based systems over abductive systems is their low discrimination ability, *i.e.* several possible diagnoses are typically suggested, with no possibility to discriminate between them. This is linked to the combinatorial nature of the algorithm, of which complexity can explode rapidly [de Kleer, 1991]. On the other hand, "abductive definitions provide good results when one can assume that the model of the system to be diagnosed is complete" [Console & Torasso, 1990]. Since it has been shown in section 2.2.1 that the knowledge of the possible faults in physical systems is typically incomplete [Davis, 1988], a consistency-based approach is adopted in this research for the construction of MVDS. Furthermore, by using only models of the correct behaviour, diagnosis of completely unknown systems or new systems is still possible. These are the reasons why this work concentrates on a consistency-based approach, since the discrimination and combinatorial explosion problems mentioned above can be helped by using the appropriate model for the physical system, as explained in section 3.

Sometimes, the use of fault models is incorporated into consistency-based methods [Struss & Dressler, 1989; de Kleer & Williams, 1989], as a way of increasing the diagnosis performance, but this subject is outside the scope of this research.

2.2.3 Consistency-based fault diagnosis

The previous section explained why a consistency-based approach is adopted for MVDS, and the definition of a consistency-based diagnosis was given. Following this approach, the diagnostic task is performed through checking for inconsistency between observed behaviour and expected behaviour obtained through the physical system's description. The process starts by assuming the normality of all the components, and then these assumptions are changed when they result in an inconsistency with the observations. Eventually, a diagnosis is a set of components such that if it is assumed that they are faulty, no more discrepancy remains between the predictions and the observations made. This approach is sometimes called the violated-expectation approach.

The violated-expectation approach has been used in the early systems such as SOPHIE [Brown *et al.*, 1982], DART [Genesereth, 1984], DEDALE [Dague *et al.*, 1987], and Davis' system [Davis, 1984]. However, the first major contributions to a general and formalised theory of consistency-based systems are GDE, by Johan de Kleer and Brian Williams [de Kleer & Williams, 1987], and DIAGNOSE, by Raymond Reiter [Reiter, 1987]. The objects and notations used in the method described in [de Kleer & Williams, 1987] have set the standards, and thus the description of GDE given below provides an explanation of a typical consistency-based approach.

2.2.3.1 General Diagnosis System (GDE), by Johan de Kleer

The article where GDE is described [de Kleer & Williams, 1987] has the following title: "Diagnosing Multiple Faults". The main advantage claimed in this article for GDE is to be able to cope with multiple faults. However, GDE's contribution to the field goes further. It consists of a general (domain independent) diagnosis strategy using qualitative reasoning in a simple common-sense algorithm.

The algorithm is organised around three important concepts, which are now adopted as common terminology and are largely used by the research community. These three concepts are:
Symptoms: A symptom is a contradiction between some predictions and some observations or test results. The discovery of a symptom is the starting point of a diagnosis task.

Conflicts: A conflict set is a set of assumptions about the correct behaviour of components out of which at least one is wrong.
Candidates: A candidate is a set of assumptions about the correct behaviour of components which, if they were all wrong, would explain all the symptoms observed.

The algorithm is composed of three main parts, namely:
- the test generation,
- the conflict recognition,
- and the candidate generation.

The basic strategy is:
- First, to find the symptoms raised from the knowledge currently available, that is to say from the command values (inputs for example) and the measurements made until then. The conflict sets related to each of these symptoms are then constructed by the *conflict recognition* module.
- Secondly, the set of the current candidates, that is to say explaining the current set of symptoms, is constructed by the *candidate generation* module.
- Thirdly, the *test generation* module generates a new test to be run, which will provide new current knowledge about the physical system. Following the iterative nature of fault-diagnosis, the process starts again from the first step, with a new run of *conflict recognition*. Indeed a new test result allows further inferences, and thereby should probably raise new contradictions, *i.e.* new symptoms, and thus new conflicts. Each modification in the set of the conflict sets leads to the second step where the candidate set is refined.

The existence of a test generation engine is an advantage. However, the test theory uses the individual probabilities of break-down for the components. It thereby uses shallow knowledge, as these probabilities are derived from recorded past experiences.

Dealing with multiple-faults has raised problems. The conflict sets and the candidates which could be constructed are too numerous. De Kleer has dealt with this complexity by working with minimal sets: a conflict set is said to be minimal if it strictly includes no conflict set, and a candidate is said to be minimal if it strictly includes no candidate set. Only minimal conflicts and minimal candidates are constructed, thus avoiding redundant pieces of information.

The qualities of GDE are numerous, beside the soundness of the performed diagnoses. The formalism is present though it is kept light, *i.e.* the workings are easily understandable, and meet the aim of common-sense reasoning. As its name General Diagnostic Engine claims, the system is able to cope with a large set of diagnostic problems, including multiple faults. The representation of the candidates and conflict sets has been kept manageable by the use of minimal sets. However, a major weakness in GDE is the absence of a framework for dealing with multiple representations, as there is in MVDS. Indeed, the physical system's model used in GDE is fixed for the whole process, and predictions obtained through this model are the basis of the process. In some applications, for example analogue circuits, an appropriate model for making predictions throughout the whole diagnostic process is not available [Dague *et al.*, 1987]. In this case, a fault-diagnosis system must be able to manage a library of models so that the appropriate model at the current stage of the task at hand is picked and used. The need for, and use of, multiple models is justified and investigated in detail in section 2.4.

GDE has therefore provided the fault-diagnosis community with a sound qualitative reasoning algorithm for consistency-based diagnosis, but the need for the ability to deal with multiple models has motivated more work, as did the need to increase the theoretical formalism behind the method. The fault-diagnosis MVDS uses GDE's algorithm for candidate generation, but manages multiple-representations. Below is described the system created by Raymond Reiter, which addresses the issue of providing GDE with a stronger formal frame.

2.2.3.2 System DIAGNOSE, by Raymond Reiter

Although the ideas are the same as in de Kleer's *GDE*, the formalism used for *DIAGNOSE* is intentionally stronger than in GDE [Reiter, 1987]. Indeed the spirit of this work is to complete de Kleer's work by constructing a more rigorous formal frame. The concept of *conflict sets* is still present, but there is no more explicit *symptom* or *candidate*. A diagnosis is constructed straight from the collection of conflict sets, through the use of abstract and mathematical tree structures. Therefore, the structure of the system consists of two modules: the main one constructs the tree structures from the conflict sets, and uses the second module that computes a conflict set when required.

The collection of conflict sets is not explicitly constructed: a conflict set satisfying some constraints is constructed, if it exists, only when required during the computation of a tree structure. The description

of the physical system consists of the models of the components (explicitly their models when they are *NORMAL*) and the connections between them, using logic sentences.

It is argued that this strict formal frame carries stiffness and pulls the reasoning away from a common-sense spirit. For example, by involving the construction and use of trees, the method has lost clarity, and stepped back towards the complexity of numerical methods.

Most of the time tree structures are used for their efficiency as a database structure. However, the database used by de Kleer has the advantage of being easily completed after a new measurement is done, thereby fitting the iterative nature of the task. As the tree-formalism introduced by Reiter does not provide a simple and efficient way to construct a new tree from an old one after one further measurement is made, adopting this formalism has increased the computational load.

The advantage of using DIAGNOSE's formal frame lies in sound definitions of the objects and sub-processes involved in the whole diagnostic process. Therefore it is easier to modify the algorithm to reach some required feature while keeping control on the effects of the modification.

However, when the diagnostic performance is required to be clearly understood by the human user, for the sake of human-computer co-operation or for the sake of an interactive learning system, the disadvantage of stepping away from common-sense reasoning is unacceptable. The system constructed in this work is required to stay close to common-sense reasoning, so that it can be studied clearly and so that human-computer co-operation is possible, as explained in section 1.2.3. Thus MVDS does not use the tree structures from DIAGNOSE. Instead, it uses GDE's algorithm for diagnostic reasoning.

2.3 Qualitative reasoning (Q.R.)

2.3.1 The motivation

Firstly, as explained in sections 1.2.2 and 1.2.3, the notion of a good model or of a good simulation depends on the available input information. Indeed, the same model cannot be reached from a large amount of knowledge or from only a few properties about a physical system. Secondly, the

appropriateness of models and simulations depends as well on their intended use. For example, if the model is to be used for a rough simulation, *i.e.* to distinguish between radically different behaviours of the physical system, then a very detailed model is useless and even detrimental for the clarity of the simulation.

For these two reasons, being able to reason with incomplete knowledge about the physical system is important. Qualitative reasoning is able to do so [Forbus, 1988], and thus is used in MVDS.

This advantage and further ones for using qualitative reasoning about physical systems are developed in this section.

Complexity of the physical systems and speed versus level of details

The complexity of physical systems was shown, in section 1.2.4, to be prohibitive for the idea of a perfect model. An imperfect model must then be used for performing valid fault-diagnosis.

In a lot of real-world situations, humans are capable of predicting in a satisfactory way the behaviour of a system. But the kind of reasoning used in this process is not the same as the one that classical physics would tend to use. A human brain does not reason with equations, nor with numerical simulations, nor again with a large amount of unnecessary detail. This reasoning, called common-sense reasoning, just deals with qualitative relations and entities, as opposed to the quantitative ones used in classical physics. This qualitative reasoning is the sort of reasoning performed by human diagnosticians, whose efficiency is known. The primary limits of the efficiency of human diagnosticians are their inability to deal with large or complex systems, and to offer fast diagnostic performances. These limits justify the need of computerised fault-diagnosis systems like MVDS.

Coming back to the level of detail issue, unnecessary detail can have a negative effect on the current task. First, because all the detail about a physical system is at various levels of importance, the irrelevant detail taken into account in a model will not reveal any interesting features about the behaviour predicted. Secondly, it will make the simulation task slower, sometimes more difficult or even impossible to be achieved [Forbus, 1984; Dague *et al.*, 1987]. As a consequence, unnecessary details should be ignored to keep the process as simple as possible.

Real-world systems and incomplete knowledge

The second characteristic of physical systems is the fact that a lot of measurements cannot be made. For example, it is not possible to measure currents in an analogue circuit, resulting in the lack of an important piece of information. As a consequence, the knowledge available about a physical system is typically incomplete: there is often a lack of exact values for parameters or input variables. This is a problem for the quantitative methods, as undertaking a numerical simulation requires a value for every parameter and every input variable of the system. On the other hand, a qualitative simulation does not need all these values, thereby allowing us to undertake a simulation with incomplete knowledge about the system.

Large scope of applications, re-usability of expressions

The way qualitative methods are constructed, and what tools they sometimes use (as for instance the bond graphs [Xia *et al.*, 1992]), allow them to be applicable to a large set of different systems, and even different domains, *e.g.* electronics or mechanical systems. For example, the quantitative relation $dH/dt = Q/A$ describes the instant variation of the height H of a liquid in a container having a diameter A at the height H, when the flow of liquid is Q. This relation only holds for the case of containers with straight and vertical sides. On the other hand, the equivalent qualitative relation $[dH/dt]=[Q]$, where the square brackets stand for "sign of", holds for almost any container [Williams, 1991].

Clear trace and explanations for verification and co-operation

The building and the use of qualitative models and of a qualitative simulation engine brings a better insight into how the system works. This allows the construction of a more efficient and, at the same time, more understandable fault-diagnosis system. Fault-diagnosis systems which use qualitative reasoning can provide the user with an appropriate trace and explanation of the ongoing process [Forbus & Falkenhainer, 1990].

This ability is necessary for a fault-diagnosis system to be a co-operative system, *i.e.* where the operator and the automated problem solver interact in order to reach a solution. This is because a human operator needs to understand an ongoing process in order to be able to interact with it. The most important advantage of co-operative problem-solving is to be able to derive a solution which is superior to that produced by either an autonomous problem-solver system or a human-solver working separately [Clarke & Smyth, 1993].

The production of an appropriate explanation also has the advantage of increasing the trust of the operator, thereby making it more likely to be used for important decision making [Dugdale, 1996; Woods, 1985].

MVDS uses qualitative reasoning for the four reasons explained above. This allows MVDS to ignore irrelevant details, to cope with incomplete knowledge about the physical system to diagnose, to have an increased scope of application and re-usability, and to produce clear explanations.

2.3.2 Ambiguity

The main drawback of qualitative reasoning is its ambiguity, resulting from not dealing with a high level of details. For example, in a simulation task, several possible behaviours can be predicted from the same set of initial values. An extra amount of work is then needed to remove this ambiguity, in order to obtain a single result.

MVDS uses qualitative reasoning despite this drawback because the advantages described in section 2.3.1 are important.

2.3.3 Formalisation of qualitative reasoning

Qualitative reasoning is formalised through the construction of specific algebras allowing manipulations and calculations on qualitative objects. A qualitative algebra typically consists of an abstraction of a quantitative one [Williams, 1991]. The two common classes of qualitative algebras are described in this section: the crisp boundaries algebras [de Kleer, 1984; de Kleer, 1987; Xia, 1993] and the order of magnitude algebras [Mavrovouniotis & Stephanopoulos, 1988; Raiman, 1988; Raiman, 1991; Parsons, 1992]. In the following sections, two square brackets around a variable, *e.g.* [A] stand for the sign of the value of the variable.

Crisp boundaries algebras

The first way of abstracting the set of real numbers is to split it into parts representing significant features of the variables. The boundaries, or landmarks, are some real numbers. For example, the set of real numbers can be abstracted to $]-\infty ; 0] \cup]0 ; +\infty[$. In this case, there is one landmark only, which is the number 0, and the qualitative abstraction of numbers would be $]-\infty ; 0]$ for any number inferior or equal to zero, and $]0 ; +\infty[$ for any number strictly superior to zero. Thus the shift from one qualitative value to another is not continuous but sharp. This is why these algebras are called the sharp boundaries algebras or crisp boundaries algebras.

The set of values that a qualitative variable can take is called the qualitative space. Usually the state of a parameter is defined by its value and its direction of change. In [de Kleer, 1987] and [de Kleer, 1984], the qualitative space considered is {negative, zero, positive}, or {-,0,+}. Then both the parameter and its direction of change take their value in this qualitative space. In [Xia, 1993], the direction of change is a value from the set {"steady", "decreasing", "increasing"}, which is equivalent to {0,-,+}, but the state itself can take its value in a more complicated set. This is because the qualitative space can be tailored to the task at hand, and in this case Xia required finer simulations than what could be obtained with {0,+,-} as the qualitative space.

Thus it is important to choose the right number of cuts in the right places, in order to obtain the relevant qualitative states and the appropriate level of details. For example, if a simulation must reveal if a certain value is below or above one hundred units, then the qualitative space {-,0,+} would be inappropriate. In this case, if the qualitative value is "-" or 0, then the quantitative value is definitely below 100 units, but if the qualitative value is "+", then it is not possible to say whether or not the quantitative value is below or above 100 units. An appropriate qualitative space for this instance is $\{\ ]-\infty;100[,\ 100,\ ]100;+\infty[\ \}$. This design is therefore domain-dependant. For example, in a system involving the temperature of a liquid, some probable landmarks are the temperature at which the liquid would boil and the temperature at which it would become solid. In another domain, these landmark values would not be appropriate.

Alternatively, the qualitative space can be designed in an undefined way, in order to stay domain-independent. For example in [Xia, 1993], the real axis is abstracted to {minimal, very-low, low, zero, high, very-high, maximal}.

After the qualitative space is designed, relations and operations between the variables must be defined. For example, in [Xia, 1993] the relations are defined via five operators:

A = B (equal qualitative states),

A= -B (opposite qualitative states),

A = der(B) (state A is the qualitative derivative of state B),

A = der(B)<=>B = int(A) (saying that state A is the qualitative derivative of state B is equivalent to saying that state B is the qualitative integral of state A),

and A = B + C (qualitative addition).

The relations built with these operators between the qualitative variables are called confluences [de Kleer & Brown, 1984]. Unfortunately, interval algebras are too ambiguous [Struss, 1987]. For example, if [A] = + and [B] = –, then [A+B] is not determined and taken to be [–∞;+∞]. This problem has led to the introduction of the order of magnitude reasoning, explained in the next section.

Order of magnitude algebras

The second way of abstracting the set of real numbers is to reason with the order of magnitude. No sharp boundary is then used. Reasoning with the order of magnitude of quantities is very common in physics, where it is a normal way of simplifying calculations. Order of magnitude reasoning is then a very natural formalisation of human scientist reasoning. This is why order of magnitude reasoning is the qualitative algebra used in MVDS. The two main sets of order of magnitude algebras are absolute or relative order of magnitude algebras.

Absolute order of magnitude

An interval arithmetic using fuzzy intervals has been studied as a base for order of magnitude reasoning. This approach is called "absolute order of magnitude" [Parsons, 1992] because the order of magnitude of a variable is not defined relatively to another variable, but directly by its distance to the fixed landmark 0. This algebra can be seen as partitioning the real numbers into an arbitrarily large set of fuzzy intervals. It provides an efficient tool for fast and clear reasoning.

Relative order of magnitude

Following this approach, a quantity is not compared to any fixed landmark, even 0. The order of magnitude of a variable is only defined relatively to another variable. There are two famous systems using this kind of algebra.

The first system is called the FOG system, developed by Olivier Raiman [Raiman, 1988]. It is based on three operators:

A *Ne* B: A is negligible with respect to B

A *Vo* B: A is close to B

A *Co* B: A is comparable to B.

These three relations, together with the axiom A *Vo* A, allow the definition of thirty one inference rules, such as:

A *Vo* B and B *Ne* C $\rightarrow$ A *Ne* C

or more complicated: { [A+B]=+, [A]= – } $\rightarrow$ { $\neg$(B *Ne* A), [B]=+ }

Raiman then designed another system, called *Estimates* [Raiman, 1991]. The system is more sophisticated and allows reasoning at different levels of accuracy. Each quantity has a qualitative value which is an interval called its coarse value. The size of this interval depends on what accuracy is required.

The second important system using relative order of magnitude is called O[M], and has been developed by M.L. Mavrovouniotis [Mavrovouniotis & Stephanopoulos, 1988]. It is based on the seven operators described below, out of which twenty-one compound relations are constructed.

A<<B A is much smaller than B

A–<B A is moderately smaller than B

A~<B A is slightly smaller than B

A==B A is exactly equal to B

A>~B A is slightly larger than B

A>–B A is moderately larger than B

A>>B A is much larger than B.

For both systems FOG and O(M), the understandability is very high, thus the common-sense reasoning aim has been achieved. Moreover, their efficiency is higher than for the interval-algebras, for instance from a decreased ambiguity [Raiman, 1991]. MVDS uses an algebra which is similar to O(M), because it is close to common-sense reasoning. This algebra is described in section 3.

2.4 Multiple representations and causal knowledge

In [Struss, 1992], the usefulness of multiple models for diagnosis is stated. It is said that these models should be appropriate for the particular diagnostic task at hand, and also appropriate with regard to the current stage of the diagnostic process. It is now widely recognised that the use of multiple models increases the diagnosis power of a fault-diagnosis system [Travé-Massuyès & Milne, 1996]. In this section, major contributions to the use of multiple-models for fault-diagnosis are described and analysed.

2.4.1 Paths of Interaction and the Locality Principle, by Randall Davis

Randall Davis' work is concerned with the use of multiple representation and causal information [Davis, 1982; Davis, 1983].

Primarily, Davis argues that two built-in assumptions about the models used by de Kleer's GDE are problematic:
- the assumption that information only flows from outputs to inputs,
- the assumption that anything not shown in the model does not exist.

Consider the former assumption. The uni-directionality of information flow is a convenient modelling tool but is argued by Davis not to be always true in fact, possibly leading thereby to false diagnoses. The latter assumption, namely the closed-world assumption, is said to prevent the detection of some faults, for example bridge faults in electronic circuits. It could be believed that, using the broadest definition possible for a fault, GDE would not have any problem to detect a bridge-fault. However, independently of the definition of a fault, GDE uses the topology of the physical system as an input. It is in the use of this physical structure information that the closed-world assumption lies, thus an alternative use of structural information is needed.

In [Davis, 1982], Davis does not give any concrete solution to address these two weaknesses, but suggests that the implicit assumptions mentioned above should be make explicitly, in order to obtain a better control of the process. Once the categories of faults of concern are decided, then surrendering or

keeping the appropriate assumptions should lead to a relevant fault-diagnosis. For example, the assumption "No wires present other than those shown" has to be surrendered if we want to deal with bridge faults. As a consequence, a diagnosis is not the only possible diagnosis, but is one which is related to the assumptions made.

In [Davis, 1983], the closed-world assumption problem is also addressed. The approach is more general and formalised than in [Davis, 1982]. The closed-world assumption is argued to be a consequence of the use of a unique representation for the physical system. In this representation, called the functional representation, the components interact with each other only through the functional paths, for example through the network of wires linking the components to each other in the case of electrical circuits. However, Davis shows that there are other ways than only through the functional path for the components to interact. This can be, for example, by the heat they produce (thermal path of interaction), or by the magnetic field they can create. These interactions, that follow non-functional paths, have an influence on the components' behaviour, and thereby are relevant in a fault-diagnosis process.

However, these non-functional paths are not present in the functional representation. Consider the thermal path of interaction. An appropriate representation of the physical system should highlight the fact that a component can interact with a certain number of other components which are located within a certain radius around it. This illustrates a first central notion in Davis' work, which is the relevance of several paths of interaction, associated with several appropriate representations of the physical system.

Consider now the path of interaction defined by the existence of unexpected wires between some components (bridges). The appropriate representation of the physical system is then the physical representation, because a component A can interact, that is to say here can be wired, with another component B if component B is physically adjacent to component A. This fact suggests that a representation ought to define the set of neighbours of each component: given a method of interaction, a component can only interact with its neighbours defined by the respective representation. This last sentence expresses the Locality Principle, the second central notion in Davis' work.

Each path of interaction defines a representation, which itself defines a new locality, that is to say an appropriate set of neighbours for each component. Following this reasoning, it appears that the reason

why bridge-faults are so difficult to diagnose with the classic fault-diagnosis systems is because two components that are neighbours in the physical representation, and thus that can be incidentally wired, are not necessarily neighbours in the functional representation that is usually used. Indeed, two physically close components are not necessarily linked to each other by design, and two components far away from each other can be wired by design.

In order to obtain a fault-diagnosis system able to diagnose any kind of fault, all the possible paths of interaction need to be considered. However, using them all at once leads to a low discrimination rate: too many components are potentially faulty. The idea is then to layer these paths of interaction [Davis, 1983]. A diagnostic process would start with the most restrictive model, taking into account a small number of paths of interaction. If the system fails at finding a candidate explaining the symptoms, a less restrictive model is then used, taking into account more numerous paths of interaction.

The importance of the concepts of multiple paths of interaction, multiple-representations, and layering of the representations has been proved. MVDS is therefore to incorporate these concepts. However, Davis' work suffers from the Single-Fault Assumption. Dealing with multiple-faults diagnosis means dealing with a set of faults which can use several different paths of interaction. The way to handle properly the multiple representations suggested by Davis, simultaneously or successively, is an important issue that adds complexity to the use of Davis' work which he has not addressed. In this work, MVDS does handle multiple representations in order to provide an efficient tool for diagnosing multiple-fault situations.

2.4.2 Peter Struss

The main subject of Peter Struss' work [Struss, 1991; Struss, 1992] is how to use multiple models, being abstractions, simplifications, and approximations of each other, within a fault-diagnosis task.

Peter Struss justifies the importance of the appropriateness of the models used for the diagnostic reasoning [Struss, 1992]: "if a diagnostic system uses a model that is inadequate for a particular case at hand, the resulting diagnosis is likely to be wrong or at least useless". However, a dilemma is identified. On the one hand, the most adequate models are usually too complex to be tractable, and on the other hand, making simplifying assumptions to make the models tractable can render them useless

in case of inadequate assumptions. The solution investigated by Struss is "to have multiple models of a system and let the diagnostic system use the one appropriate for the particular case. It should also be appropriate with respect to the stage of the diagnostic process". In other words two classes of criteria are relevant for the selection of the model which is to be used. One class is concerned with the particular diagnostic case at hand, *i.e.* concerned with which modelling simplifications are valid for the investigated system. The other class is concerned with the current state of progress in the ongoing diagnostic process, i.e. these criteria are used to decide whether the current system's representation should be exchanged for another. The work in this thesis is founded on the same justifications for using multiple models, and focuses on the second class of criteria cited above, which is concerned with the current state of the diagnostic process. The investigation of simplifying assumptions which hold before the start of the diagnostic process is not within the scope of this research.

Peter Struss gives two justifications for the relevance of adapting the model during the diagnostic process. First, what model is required for the particular case is not obvious from the beginning. Second, different stages of the process might require different models [Struss, 1992]. This is why MVDS revises the appropriateness of the model regularly through the diagnostic process, in order to assure that the model required at the current stage of the process is used.

Another issue is what models should be available for selection. In [Struss, 1991; Struss, 1992], the multiple models are abstractions, simplifications, and approximations of each other. Peter Struss argues that the typically informal use of these notions has led to confusion. As a consequence, he suggests formal definitions of these concepts. Below are the ideas behind these definitions, of which mathematical versions are not recalled here. See [Struss, 1991] for details.

Abstraction: this is a change in the representation used for a model. It stays defined by the same relations. For example, the use of qualitative values is an abstraction: while the relations between the variables are the same, these variables are represented by their qualitative value (for instance their sign).

Simplification: this is a change of the relations, based on the same representation. The variables stay the same, but the equations (as an instance of relations) linking them are changed.

Approximation: this is a special case of simplification. The idea is that an approximation of a model is a simplification of it, which must be close enough to it. The formal definition of "close enough" requires the representation to be a metric space.

The several models obtained by these simplifications, abstractions, and approximations, are organised in a tree called a model-graph. At the beginning of the diagnostic task, the roughest models are used. A progression in the model graph towards more precise models is done when a revision of the modelling assumptions and/or a refinement of models is required. This exchange from simple models to more complete models is a management issue of the model-graph. In this research, MVDS adopts the same method which consists of increasing the completeness of the models as the process goes on.

However, major differences exist between the strategy from Peter Struss and MVDS:

- The model-graph includes correct models as well as models of faulty behaviours, as opposed to MVDS which only uses correct models, as stated in section 2.2.2.

- The models available for use in MVDS are not abstractions, simplifications, or approximations of each other. They are models of the behaviour of different entities involved in the physical system's behaviour. This is because this organisation of the set of models based on the different entities is appropriate for the specific task of diagnosing complex multiple-fault situations, as explained in section 3 about the design of MVDS.

- The criteria for deciding to change the model, and which one to change to, are different in MVDS and in Peter Struss' work. In Struss' system, a more complete model is used in place of the current one when using the latter results in under-discrimination. In MVDS, a further entity is added in the system's representation when causal information indicates that this entity may be involved in a fault.

An example used by Struss [Struss, 1991] deals with an electronic component, namely a thyristor, which is made of an anode, a cathode, and a gate. This is a small set of components, and when dealing with physical real-world systems, the complexity will be of another order of magnitude. Facing this higher complexity, Struss' formalism might be difficult to use. Indeed the amount of knowledge required about the physical system would probably not be available, or would be costly to obtain. In general, although a model-graph is a powerful tool, its construction is very demanding in terms of knowledge about the physical system and in terms of modelling work. This is opposed to the desire for a common-sense system working on partial knowledge. Addressing this issue of keeping the strategy

close to an understandable process, MVDS does not require a highly formal structure for the set of models. The set of models to be used with MVDS is therefore more easily constructed, even on complex physical systems. This is because the distinction between the models in MVDS is founded on a physical distinction (the different physical entities) which can be dealt with without much formalism.

2.4.3 Oskar Dressler *et al.*

In [Dressler *et al.*, 1993], multiple representations are used by a fault-diagnosis system applied to the diagnosis of ballast tank systems, as for example in offshore plants. The system is a consistency-based system, using fault-models. The system description (SD) contains, for each component, a quantitative model and a qualitative model. The latter model is obtained as an abstraction of the former one, in the sense of Struss' definition of abstractions [Struss, 1992].

Monitoring and fault-diagnosis is performed in the following manner. Only the qualitative models are used for monitoring. When this process detects a fault, the fault-diagnosis process is launched, again using only the qualitative models. If this process results in two diagnoses which cannot be discriminated, then the qualitative models that have not yet produced a refuted prediction are switched to their quantitative correspondent. The refined predictions obtained are then likely to reveal further conflicts which result in discrimination between the previously obtained candidates. However, the use of model exchanges described by Dressler *et al.* is restricted to one switch only, in response to an under-discrimination of the candidates. This technique could be added to any fault-diagnosis system using abstracted models in order to address a discrimination problem at the end of the process, but is not a general method for the use of multiple representations.

Similar to Dressler's system, model exchanges in MVDS allow the use of simple models while they are sufficient, and the use of more complete models when they are required. However, MVDS addresses a more complete strategy for adapting models to the current stage of the process, where a series of more than one change would be possible before reaching the final result.

A quality of Dressler's method is the re-use of previous intermediate results after a model switch is performed. After some qualitative models are switched to quantitative models, the predictions obtained with the latter are combined with the predictions previously obtained on the unchanged

qualitative models. This re-use of previous predictions, as opposed to a complete re-start of the process, after some models are modified, is also an important feature of MVDS.

2.4.4 System CA-EN, by Louise Travé-Massuyès and Robert Milne

A condition-monitoring system called TIGER has been developed by a European team including Louise Travé-Massuyès and Robert Milne [Milne *et al.*, 1996; Travé-Massuyès & Milne, 1996; Travé-Massuyès & Milne, 1997]. This system performs monitoring, fault-detection, and fault-diagnosis of physical systems, and has been implemented to monitoring EXXON gas-turbines in Scotland. The part of TIGER which performs fault-detection and fault-diagnosis is called the CA-EN system. This system does not include the use of multiple representations in the process, but it makes an interesting use of causal knowledge.

CA-EN is a consistency-based method. The diagnosis algorithm is inspired by Reiter's algorithm [Reiter, 1987], with the use of hitting sets. This use of Reiter's formalism is a major difference between CA-EN and MVDS, since the original algorithm from de Kleer [de Kleer & Brown, 1987] is used in MVDS. However, Reiter's and de Kleer's algorithms have in common the use of a system description (SD), which therefore needs to be specified in MVDS and in CA-EN. The originality of this latter system is to combine the use of empirical causal knowledge and first principles, by incorporating both in the causal graph used as the SD.

This inclusion of empirical knowledge in the SD is a risk, since empirical knowledge is rarely certain knowledge. Thus, this inclusion could result in an invalid diagnosis. As a consequence, MVDS uses a SD where empirical knowledge is not included. The empirical knowledge used in MVDS only acts as a guide to the process and is not critical to the reliability of the system.

2.5 Conclusion

Model-based systems are preferred to rule-based systems, because of these latter systems' decisive disadvantages, which are their inability to produce useful explanations, their low performance on diagnosis of unusual faults, and their subjectivity.

Between the two classes of model-based systems, which are the consistency-based systems and the abductive systems, the focus is on the former class, because of their better performance with incomplete knowledge. The disadvantage of the consistency-based systems is their low discrimination power. This can be helped by using an appropriate system's representation. Hence, a consistency-based fault-diagnosis strategy is a valid research direction.
The two seminal diagnosis systems using this approach are GDE, by de Kleer and Williams, and DIAGNOSE, by Reiter.

Qualitative reasoning is used for MVDS because it copes with the complexity of physical systems by ignoring unnecessary details, because it can be undertaken with incomplete knowledge, because it allows the construction of more re-usable systems, and because it enables the production of useful traces and explanations. The different qualitative algebras are the crisp boundaries algebras and the order of magnitude algebras. Within these latter algebras, there are absolute order of magnitude algebras and relative order of magnitude algebras. The algebra used for MVDS is required to be close to common-sense reasoning, in order to optimise the advantage of using qualitative reasoning. This is why the algebra chosen is a relative order of magnitude algebra.

Finally, the use of multiple representations and use of causal knowledge in diagnosis systems have been examined. The study of Randall Davis' work has shown that they are closely related, since multiple models are defined by multiple paths of causal interaction. Peter Struss stresses the importance of using multiple models and uses a model-graph where they are logically related in order to organise their use. However, a different set of models, using different entities, would be more appropriate for diagnosing complex multiple-fault situations. Furthermore, the construction of a model-graph as in Struss' formalism is expensive and not always possible. Oskar Dressler and his colleagues have demonstrated the use of multiple models on an industrial application, but this use is restricted by the fact that only one change can occur. Louise Travé-Massuyès and Robert Milne do not make use of multiple representations but integrate the uses of empirical causal knowledge and deep-

knowledge. However, this use is risky, since the validity of the diagnosis depends on the soundness of the empirical knowledge.

From the study of the advantages and drawbacks of these researchers' works, six points have been raised which are addressed by MVDS:

- multiple-representations must be used for the diagnosis of multiple-fault situations,
- the models used for diagnosis are adapted to the current stage of the diagnostic process.
- the set of models available for use at any stage in the diagnosis process is appropriate for the specific problem addressed, *i.e.* complex multiple-fault situations,
- the model changes are allowed to happen once or several times, depending on the diagnosis process at hand,
- after a model change occurs, the process is not re-started from the beginning but it continues, using some previously made predictions,
- integrating empirical knowledge is enabled, but the use of this inherently uncertain knowledge is not critical to the correctness of the result.

In the next chapter, these issues are considered with respect to the domain of application. It is shown how they guided the design of MVDS.

Chapter 3

Designing MVDS

3.1 Introduction

It has been shown in chapter 2 that the existing consistency-based fault-diagnosis systems suffer from some restrictions in their use of multiple-representations. A more powerful use of multiple-representations is therefore an aim of MVDS' design described in this chapter. A second aim is to use this tool of multiple representations for the diagnosis of specific multiple-fault situations called complex multiple-fault situations.

Section 3.2 defines the problem of diagnosing a complex multiple-fault situation, and describes the characteristics of physical systems that make them prone to developing these situations. Four of these characteristics have been identified: being non-monitored (or ill-monitored), needing regular maintenance, including partially unknown components, and being subject to variable and partially unknown conditions of use.

It is shown in section 3.3 that all these characteristics are present in water-distribution systems, a description of which is presented in the same section. Thus, these systems are an appropriate test-bed for MVDS.

Section 3.4 explains the design of MVDS and how it results from the requirements for diagnosing physical systems in general and complex multiple-fault situations in particular.

3.2 Complex multiple-fault situations

The importance of multiple-faults in physical systems has been introduced in section 1.4. It has been related to the presence of numerous entities, whose behaviours influence each other, thereby spreading faults across the system. Because of these influences across different entities of the system, it is argued that the multiple faults are likely to involve several entities. This is illustrated in the example presented in the following section.

3.2.1 A motivating example

Consider the example of the 3-piece air-conditioning system pictured in figure 3.1 below. Its behaviour involves at least three entities: the flow of air, the temperature of air, and the air dust content. Saying that several entities are involved in a multiple-fault situation means, for example, that the dust filter is failing so that the dust content in the system is too high, and that the heater is also failing, so that the temperature of the air output is too low. The fault-situation involves then two entities, the dust content and the temperature.

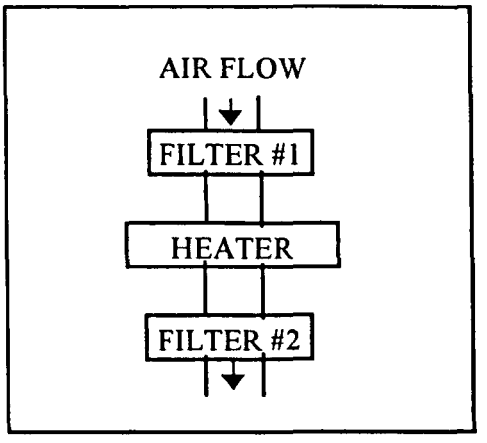


Figure 3.1 - 3-piece air-conditioning system

3.2.2 Diagnosing complex multiple-fault situations

The fault scenario from the example above contains a situation called a multiple-fault situation, as introduced in section 1.5.

Definition (Complex multiple-fault situation)
A situation where several faults are present, and where these faults affect several entities, is called a complex multiple-fault situation.

Obviously, diagnosing a complex multiple-fault situation requires that all the entities involved are present in the system's representation. In the previous example, if the dust content was not included in the system's representation, then the fault resulting in a too high dust content could not be found. As a consequence, attention must be paid to integrating the necessary entities in the system's representation.

On the other hand, integrating many of these entities increases the cost of the diagnosis process in two ways. The larger the system's representation, the higher the computational cost and the larger the number of measurements needed to perform the computations.

As a consequence, the first solution to diagnosing complex multiple-fault situations, which would be to integrate the largest set of entities available into the system's representation, is not viable. The cost in measurements would be too high for many systems where measurements are expensive in time or money, and the risk of a combinatorial explosion would be increased.

The solution proposed and investigated in this research is to avoid integrating all the available models of entities into the system's representation at once, as described in section 3.2.3.

3.2.3 Criteria of complex multiple-fault situations likelihood

MVDS is specifically constructed, and therefore should be especially used, for diagnosing complex multiple-fault situations, even if its range of use is not restricted to them. Therefore, it is useful to be

able to identify if a system is prone to these situations, and thus decide if MVDS is the most appropriate strategy for diagnosing the system.

It is argued that some characteristics of a physical system make these situations more likely to happen. Four characteristics have been identified and are described below. They are related to being non-monitored (or poorly monitored), requiring some maintenance work, involving components which are insufficiently known, and being used in variable and uncontrolled conditions.

3.2.3.1 Non-monitored systems

Because in non-monitored systems a fault can occur and not be spotted for some time, and because some faults do not prevent the whole system from working (at least for a while), it is possible that a system is kept functioning when it contains one or more faulty components. During that time, the other components function under abnormal conditions, due to the primary faults. These functioning under abnormal conditions can result in other components becoming faulty. When the acknowledgement that a fault is present occurs, it is possible that the fault which alerted the user is only the tail-end of a chain of faults.

Obviously, when a fault occurs in an appropriately monitored system, it should be promptly noticed and corrective action should then be taken, so that no other fault results from the abnormal influence of the primary fault. However, this prompt notice is not always present, and therefore any monitored system where the reaction time is inappropriately long could be subject to a chain of faults and could thus benefit from being diagnosed by MVDS.

3.2.3.2 Maintenance requirements

If a physical system requires maintenance, then the potential for maintenance faults is present. They are important because if a component needs regular maintenance, it means that its functionality ages quickly (compared to the ageing speed of the system as a whole), and therefore improper maintenance could result in fast and important changes in the component's behaviour. It could either work less efficiently, work in a different manner, or even completely stop working.

Consider a first example [Chadderton, 1991]. Steam boilers accumulate limescale from the water passing through the treatment plant. This accumulation must be prevented from becoming too important, as accumulated salts could be carried over into the steam pipes and clog safety valves. It is an instance of a system in which functioning can be seriously disturbed by limestone and other minerals contained in the water. Devices for softening the water are used in conjunction with these machines. The water treatment can be based on chemical reactions. The water is forced to flow through a tank of a chemical, called sodium zeolite, so that, as it comes out, the flow only contains some harmless sodium carbonate, a non-scale-forming salt. The reaction is:

calcium carbonite (in water) + sodium zeolite (in tank)
→ sodium carbonate (in water) + calcium zeolite (in tank).

When the device is used, it accumulates some calcium zeolite and its reserve of sodium zeolite decreases. As a consequence the device demands two regular acts of maintenance. It must be re-loaded with sodium zeolite, and the accumulated calcium zeolite must be disposed. Failure in maintaining the water softener can result in a dramatic blockage of pipes which might be discovered quickly. However, it may only result in a decrease in the device's performance, with some small quantity of scale-forming salt contaminating the output water flow. This amount of scale-forming salt would be higher than normal, but not high enough to affect the functioning of the system in an immediately noticeable manner. Therefore parts of the physical system can be slowly affected by the abnormal water flow composition, and the potential exists for the creation of a chain of faults.

It is argued that the result of improper maintenance is typically a progressive deviation of behaviour, which has a negative and destructive effect on the system. It can be seen as a small fault which progressively becomes more important. It is this progressive aspect of maintenance-related faults which gives room for the formation of a chain of faults, thus facilitating complex multiple-fault situations.

3.2.3.3 Partially known components

The incompleteness of the knowledge available about a component's behaviour and ageing process is another factor that needs to be examined. Consider a spring made by a high technology company and used in an electronic device for dampening possible vibrations. In this case the component is well

known. Exact equations are available for the precise simulation of the spring's movements, and the ageing of the spring is also a well known process. On the other hand, consider a pipe in a plumbing system. The component itself is only partially known, for example it is not exactly clear what chemical reactions are actually happening between the pipe's wall and the inside flow.

Some unknown features of this component's behaviour may be harmful to other components, and because this harmful action is not known, nothing will be done about it before a fault is discovered. During this time a complex multiple-fault situation might have developed. For another example, consider a suspension bridge. Assume that the suspension cables react to a strong wind by vibrating in a manner which was not expected by the engineer. It means that the knowledge about these cables' behaviour was only partial. This unexpected behaviour can result in an unexpected amount of vibration in the bridge's main body, which it may not have been constructed to resist. Parts in the bridge's body may then contain cracks, which modify their own dynamic behaviour, i.e. these parts are now faulty. These faults may not be discovered before the whole bridge actually collapses.

3.2.3.4 Variable conditions of use and unknown conditions of use

Compare, as above, the use of a spring in an electronic device and the use of a pipe in a plumbing system. The changes in the condition of use of the spring are minimal. An electronic device is normally not subject to widely varied conditions of use, and any possible deviations are usually known. For example, it is known qualitatively and quantitatively with a sufficient accuracy how the ambient temperature affects the spring. On the other hand, the conditions of use of a plumbing pipe can have large variations of different sorts. The temperature of the flow can be from very cold to very hot, the temperature around the pipe can span a large spectrum of values, the composition of the liquid can include many different elements and each element's concentration can have large fluctuations.

Furthermore, even in the case of steady conditions of use, they are sometimes not entirely known. For example, this is the case for the exact composition of the fluids that might sometimes flow through the pipe mentioned above, or the temperature that these liquids may have. As well as ignoring the exact behaviour of the pipe under these unknown conditions, another consequence of this partial knowledge is the ignorance of the current state of repair and ongoing ageing process of the pipe. This lack of knowledge results in the prolonged use of the system without knowing exactly its state of repair nor its conditions of use. Complex multiple-fault situations can then develop during this period of use.

Identifying some of these characteristics in a system means that complex multiple-fault situations are likely to develop, and therefore it is useful to employ MVDS for diagnosing faults in the system. The way MVDS addresses this task is through the use of multiple representations, as explained in the next section.

3.2.4 Multiple-representations to tackle complex multiple-fault situations

Because numerous entities are potentially of concern in a system's diagnostic process and because their investigation is costly, the set of investigated entities needs to be kept to a minimum. This is why the use of multiple representations is the solution adopted in this research.

In order to keep the system's representation to a minimum size, MVDS integrates an entity in this representation only once there is evidence that this entity is potentially involved in a fault. The system's representation therefore first incorporates the only entity which is known to be related to a fault: the entity which revealed a malfunction and started the diagnostic process. As explained above, investigating only one entity cannot diagnose complex multiple-fault situations, and therefore a system's representation integrating more entities needs to be considered at some later stage.

The need for the use of multiple representations has been identified as a relevant manner to address the diagnosis of complex multiple-fault situations. Recall that the use of multiple representations is also widely recognised to be useful for diagnosis systems in general [Struss, 1992], as described in section 2.4.

In order to undertake these changes of representation, MVDS needs to be able to decide:
- when to undertake a change,
- which change to undertake.

These two issues are investigated in section 3.4.2.2.

Furthermore, the set of models, *i.e.* the set of the different representations available, needs to be designed. What are the different models that need to be available in order to perform the diagnosis of complex multiple-fault situations? This issue is addressed in section 3.4.2.1.

3.3 Domestic water distribution systems

3.3.1 General presentation

The focus is placed on hot water and heating systems. These systems are common-place (as virtually every building in developed countries includes one). Their size and complexity varies widely, from small domestic systems to industrial systems and even district heating systems. For the sake of clarity, the components and system studied in this research correspond to a small domestic system. However, as will be explained in chapter 5, MVDS copes with larger systems and could therefore be applied to industrial or district systems.

Before going any further into the investigation of a hot-water and heating system as a case-study for MVDS, some plumbing terms are introduced and the non-triviality of this type of system is highlighted.

In appropriate plumbing terms, domestic hot water (DHW) relates to the water supplied for washing purposes. "If the washing process is required for some industrial process, the water may have to be chemically treated in order to protect the process from any salts or metals present in the water supply. In the domestic and commercial field the supply of hot water is normally for body, clothes and dish washing" [Curd & Howard, 1996]. This mention of the use of hot water for an industrial process, and the precautions that need to be taken about what is contained in the water supply, is a first hint that hot water distribution systems are not as simple as one may think. This is reinforced by the following: "the convenience of piped water systems is likely to be taken as granted by those who have not been camping or caravaning. The provision of safe and hygienic water supplies is of paramount importance, and a considerable amount of engineering is involved in such provision" [Chadderton, 1991]. This quote stresses the potential complexity of water systems and also points out the importance of reliable systems. The computerised diagnosis of these systems is therefore a relevant task, for ensuring efficient repair and efficient restarting of failed systems.

Furthermore, "hot water supply cannot be considered in isolation from central heating because systems commonly combine both functions" [Garrett, 1991]. This increases the typical complexity of hot water systems. This type of system, combining hot water supply and central heating, is the case-study used in this research.

The typical components used in a domestic system are:
- a boiler, which heats up cold water feeds,
- a pump, which ensures an efficient circulation of the water in the system. This is necessary in large buildings, where it would be impossible to get the water to circulate the whole system because of gravity,
- radiators (or heat emitters), which ensure an efficient transfer of the heat from the water to the surrounding air,
- a hot water vessel, which is a container where water is heated and stored, before being used for washing purposes (in some less usual cases for drinking),
- a water softener, which reduces scale formation by changing scale forming salts into non-scale-forming salts.

These components involve the following significant entities in their behaviours:
- the water pressure,
- the water flow,
- the water temperature,
- the water air content,
- the water hardness.

A more detailed description of these components and entities is given in chapter 5 where the case-study is explained.

3.3.2 An appropriate test-bed for MVDS

Hot water and central heating systems make an appropriate test-bed for MVDS because they are prone to complex multiple-fault situations.

First of all, there is a significant number of entities that are of importance for the system's behaviour, in the sense that if the entity is affected, then the whole system can be considered to be faulty. In other words, the whole system is functioning correctly only if the flow along the system, the pressure, the hardness of the water, its temperature and the air content, are all behaving correctly. The set of potential faults is therefore related to a large set of entities.

Secondly, the causal influences between these entities are numerous. Independently of any specific system, the following influences can be identified:
- the water flow and the water pressure are directly related,
- the water hardness results in scale deposits and therefore influences the flow (this is particularly noticeable in small pipes),
- for the same reason as above, water hardness influences the efficiency of pumps,
- the flow in the radiators influences the temperature of the water, since a slow flow will cool the water,
- the air content in the water influences the efficiency of pumps,
- the air content in the water influences the flow because a high air content allows the creation of "sludge", a deposit that perturbs the water flow.

The two paragraphs above have established that water distribution systems are complex enough to offer interesting fault-diagnosis problems, and that there is potential for the occurrence of complex multiple-fault situations. However, it is necessary to establish if these possible complex multiple-fault situations are actually likely or unlikely to develop. This is undertaken through the evaluation of the four characteristics described in section 3.2.3.

These four characteristics are all present in water distribution systems:

- Most of the time, they are not monitored. This is the case in all domestic systems.
- Maintenance is required for radiators, that need to be bled regularly. Maintenance is also required for water softeners, since the salts they contain need to be replaced regularly.
- The behaviour and ageing process of many components are typically only partially known. For example, how smooth the water flow is inside a radiator, given that sludge and scale deposits may be present, is not usually known. Similarly, the actual performance of a pump, after a certain number of hours of service, is not precisely known.

- The conditions of use vary widely, and are not entirely known or controlled. For example, the hardness, temperature, pressure and air content of the main water supply may all vary significantly without warning or evidence.

Because of all these characteristics, water distribution systems are prone to develop complex multiple-fault situations. This is why a water-distribution system is used as a case-study in this research. The next section investigates how MVDS is designed, for the task of diagnosing physical systems in general and water-distribution systems in particular.

3.4 Designing MVDS

3.4.1 General requirements for physical systems

3.4.1.1 Coping with incomplete knowledge: the qualitative algebra

Knowledge about physical systems is incomplete, as explained in section 1.3.4. The use of qualitative reasoning allows reasoning even with this incompleteness, as seen in section 2.3. Thus, MVDS uses qualitative reasoning for the whole diagnostic process. Similar to the algebra used in O[M] [Mavrovouniotis & Stephanopoulos, 1988], the algebra used is based on five qualitative states, five operators and fifteen rules.

The five qualitative states are 'very_high', 'high', 'medium', 'low', and 'very_low', and the operators are:
>> , standing for "is much greater than",
~> , standing for "is slightly greater than",
== , standing for "is equal to",
~< , standing for "is slightly smaller than",
<< , standing for "is much smaller than".

The relations between the qualitative states are defined by the following rules:

rule(very_low,<<,medium).

rule(low,<<,high).

rule(medium,<<,very_high).

rule(very_low, ~<,low).

rule(low, ~<,medium).

rule(medium, ~<, high).

rule(high, ~<, very_high).

rule(L, ==, L). This means that the relation "==" stands between any two identical objects.

rule(very_high, ~>, high).

rule(high, ~>, medium).

rule(medium, ~>, low).

rule(low, ~>, very_low).

rule(very_high, >>, medium).

rule(high,>>,low).

rule(medium,>>,very_low).

This algebra allows reasoning with incomplete knowledge about the models and incomplete knowledge about the current behaviour.

3.4.1.2 Coping with complexity

Diagnosing complex physical systems is addressed by MVDS in several ways.

- it uses the qualitative algebra described in the previous section. This is how unnecessary detail is discarded from the reasoning, as explained in section 2.3.1.
- it minimises the size of the model used, in order to minimise the cost of making predictions. This minimisation is obtained through the use of multiple representations described in section 1.3.5.

In order to reason about the fault-diagnosis task as an iterative process [Reggia et al., 1984], the term *diagnosis session* is useful. It refers to the process of obtaining diagnosis candidates without having modified either the system's representation or the set of observations between the start and the end of the process. The use of multiple representations is obtained by designing MVDS as a series of diagnosis sessions, where changes in the system's representation occur in between the diagnosis sessions. In this way, the system's representation can be adapted during the diagnostic process to fit as closely as possible to the most efficient model.

The detail of the management of the multiple representation, *i.e.* how the system's representation is changed between the diagnosis sessions, is explained in section 3.4.2.3.

3.4.1.3 Producing trace and explanations

The possibility of producing useful traces and explanations of the diagnostic process stems from the use of qualitative reasoning. A detailed trace of each diagnostic process is argued to be useful for human supervision, human/computer co-operation, and is even argued to be a potential sufficient alternative to explanation. In other words, if the trace produced includes the intermediate results obtained and actions taken along the process, then it includes an explanation of the diagnostic reasoning. One advantage of this method over printing a less complete trace and then producing an explanation, is that the task of a-posterior building of an explanation is avoided. The second advantage is that, in the case of a co-operative process, explanation about the intermediate stages of the process is always available. However, a trace only provides an explanation of how intermediate results and the final solution have been reached. Producing an explanation of the occurring faults (possibly together with why a particular repair is needed) is possible, but it is not undertaken in the current format of MVDS.

As explained in section 2.2.1, rule-based systems are unable to produce meaningful explanations [Torasso & Console, 1989]. This is why a model-based approach is used for MVDS. More precisely, as explained in section 2.2.2, a consistency-based approach has been chosen over abductive approaches, because no fault-model is needed with a consistency-based approach. Between the two main algorithms, namely GDE and DIAGNOSE, GDE is the most appropriate algorithm to be used in MVDS, because of its simplicity. GDE uses a common-sense method, where no abstract objects (such as trees, used in DIAGNOSE) are present. Therefore, a complete trace of a process following GDE's

method is suitable as an explanation in the sense discussed in the previous paragraph. There is no need for a translation phase from abstract concepts towards the common-sense notions needed in an explanation.

The rich trace provided by MVDS' output includes the following features.
- It recalls the reasoning hypothesis, e.g. what is the current model used and what are the measurements available for checking consistency.
- It lists the predictions made. For each of them, the display includes the value predicted, the entity, the location in the system, and the set of components of which models have been used to reach this prediction.
- It lists the symptoms constructed from the latest batch of predictions.
- It displays the minimal conflict sets, taking into account the entire set of symptoms constructed since the start of the process.
- It displays the minimal candidates, taking into account the entire set of minimal conflict sets constructed since the start of the process.
- It provides also a precise account of the reasoning undertaken in relation to the management of the multiple representations. This management and its trace are described in section 3.4.2.2 below.

In addition, the output from MVDS is fully structured with English sentences. Thus, the operator can understand clearly the diagnostic reasoning, thereby easing human-checking and human co-operation.

3.4.1.4 Enabling co-operative work

The production of a clear and detailed trace by MVDS is an important step in enabling co-operation with an operator. Thanks to this trace, the operator has an appropriate understanding of the ongoing process, and thus can judge whether or not to intervene. In the case of a decision to intervene, MVDS' design needs to enable this intervention.

There are several kinds of possible interventions. Because co-operation is not the central matter of this research, the set of possible interventions in MVDS has been limited to a single intervention. However, this single intervention suffices to demonstrate the possibility of co-operation and its usefulness.

This intervention is as follows: when the operator believes that the process should stop, then he/she has the opportunity to have it stopped. Here is a possible reason for the operator to believe that the process can be ended: one entity E, and only one, is suggested for investigation as a result of the latest dynamic revision, and the operator has some knowledge that makes him/her believe that no fault is affecting this entity. In this case, no further investigations are needed.

The resulting feature of MVDS, allowing this intervention, is a break of the process after each dynamic revision, while the operator is prompted to indicate whether or not the process must continue.

3.4.2 Specific requirements for diagnosing complex multiple-fault situations

3.4.2.1 The appropriate set of models

So that MVDS can investigate the smallest set of entities needed by using multiple representations, these representations need to be tailored with respect to the set of entities they incorporate. This is achieved in MVDS by making independent models of the entities available. For each relevant entity in the physical system's behaviour, a model describes the correct behaviour of this entity in the system. Predictions can then be done through only one of these models or through any number of them. Changing the system's representation is, in fact, changing the number of models that are used for making predictions.

Obtaining models which are independent of each other is possible either by using functional modelling [Lind, 1994] or by modelling the steady state behaviour of components, thus allowing the decoupling of the entities.

Considering independent models for the different entities has two advantages every time one further entity is to be investigated:
- there is no need to re-model the system taking the entity into account.
- the process does not start from scratch again, as all the predictions and conflict sets obtained with the previous set of entities are still valid.

Using sets of independent models of the entities for the system's representations allows a control in the number of entities involved in the system's representation whilst keeping the process economical. This is possible since no modelling work needs to be done when a representation's change is decided and the previously made predictions are also still valid. The set of models is managed in the way described in the next section.

3.4.2.2 Use of causal knowledge

As explained in section 1.4, the multiple-fault situations happen because of the influences existing between quantities and entities. MVDS is designed to use the causal knowledge, containing the information about these influences in order to follow the influence path that has led to the multiple-fault situation. If this path is followed completely, then no entity affected by the multiple-fault situation will be absent from the diagnosis. In this case, the resulting diagnosis is said to be entity-complete, as defined below.

Definition (entity-complete diagnosis):
A diagnosis is said entity-complete if it has been obtained through a process where each entity affected by a fault has been investigated.

Obtaining an entity-complete diagnosis is the aim of MVDS. However this result requires the causal knowledge to include all the influences that are responsible for the current multiple-fault situation.

Result:
MVDS results in an entity-complete diagnosis if the causal knowledge includes all the influences that have caused the current complex multiple-fault situation.

Note that an entity-complete diagnosis is not necessarily a perfect diagnosis. The usual problem of having enough measurements for a complete discrimination between diagnosis candidates still holds.

The causal knowledge used in MVDS is a set of influences. MVDS distinguishes between two types of influences. A local influence is an influence between two quantities, whereas a global influence is between two entities, everywhere in the system. For example, between any two points in an electrical circuit, the intensity of current is always influenced by the gradient of voltages: this is a global

influence between current intensity and gradient of voltages. In contrast, in the air conditioning system, the influence of the content of dust over the temperature of air is only valid within the frame of the heater. This influence does not exist in the filters for example. It is therefore a local influence. The overall set of influences can be represented graphically by the *graph of causal dependencies*, where quantities are the nodes, and influences are arrows between the quantity boxes. The local influences are represented by one arrow between two quantities, and the global influences as a number of arrows linking the same couple of quantities in every component.

Figure 3.2 below is a possible graph of causal dependencies for the air-conditioning example described in section 3.2.1.

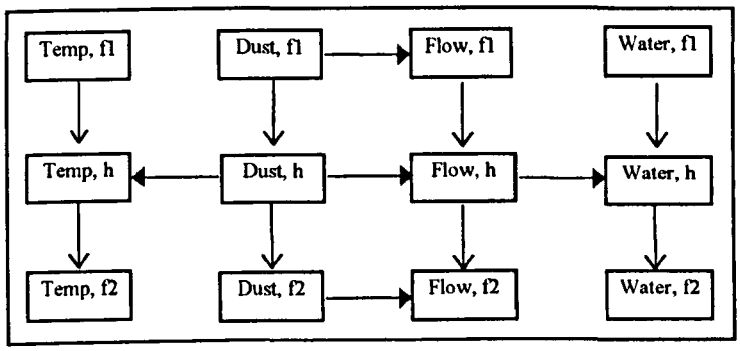


Figure 3.2 - Graph of causal dependencies for the air-conditioning example

The term Temp stands for Temperature, h for heater, f1 for filter#1, and f2 for filter#2. This graph represents the following influences:

- a local influence between the temperature and the dust within the heater,
- a global influence between the dust and the flow,
- a local influence between the flow and the water within the heater.

The causal knowledge containing the local and global influences is used by MVDS for guiding the management of the system's representation. This guidance occurs whenever the system's representation is considered for a modification, within the process of a dynamic revision, as described in the next section.

3.4.2.3 Dynamic revisions

As explained in section 3.4.1.2, MVDS is designed as a series of diagnosis sessions and changes of system's representations. These changes are performed through a process called a dynamic revision.

First of all, note that the set of entities that are currently used for making predictions is called the set of entities *under focus*. After every diagnosis session, a dynamic revision selects what entities are potentially affected by a fault, and should therefore be under focus. These entities are the ones which are causally linked with an entity that revealed some conflict sets in the latest diagnosis session. The causal knowledge available in MVDS is therefore necessary for performing a dynamic revision.

In order to ensure that the best change is performed by a dynamic revision, it takes into account the latest predictions and resulting conflict sets. This use of the latest computed results is the reason why these changes are qualified as 'dynamic'.

Having designed MVDS as a series of diagnosis sessions and dynamic revisions, and having the set of models designed in the manner explained above, it is possible to include in the system's representation all the entities that are potentially involved in the faulty situation. Furthermore, deciding to make these inclusions can be based on the most recently updated knowledge about this faulty situation. The latest intermediate diagnosis results and the causal knowledge are the two types of information used within a dynamic revision.

3.5 Conclusion

This chapter has described the study of the general constraints in diagnosing physical systems, the study of more specific constraints about diagnosing complex multiple-fault situations, and the consequences of all these constraints on MVDS' design.

The diagnosis of these latter situations has been identified in section 3.2 as a task where care must be taken about which entities of a physical system are investigated. This is because a lack of care in selecting these entities would result in either an inaccurate diagnosis, or an overload of measurement and computational work.

In order to identify physical systems that are prone to develop complex multiple-fault situations, and which therefore would benefit from being diagnosed by MVDS, four criteria have been described:

- being non-monitored,

- needing regular maintenance work,
- integrating partially known components,
- being subject to variable and partially known conditions of use.

Because water distribution systems fulfilled all these criteria, they have been identified as a class of systems at risk. Thus, they provide an appropriate test-bed for MVDS.

In section 4 of this chapter, the design of MVDS was described and justified. A qualitative algebra is used for coping with incomplete knowledge and with the complexity of the system. The production of a rich trace is made possible through the use of the qualitative algebra and through using GDE's method for diagnostic reasoning. In order to have a co-operative system, the strategy includes a break in the process where the operator can intervene. The set of models in the library is structured into independent models of entities, and causal knowledge is used for guiding the management of the system's representation. Finally, dynamic revisions are included in the process, undertaking changes in the system's representation in order to adapt it to the current state of the process.

The implementation of these design features is described in chapter 4, together with MVDS' algorithm.

Chapter 4

MVDS' algorithm and implementation

4.1 Introduction

The design of MVDS, resulting from several task requirements, has been described in chapter 3. Its implementation is the subject of this chapter.

In the first section, the various objects used in the implementation of MVDS are described. They are separated into two groups, depending on whether they are used for the description of the physical system at hand or for the expression of the components' models in the library.

The conceptual architecture of MVDS is explained in section 4.3. It describes the four reasoning modules, namely the diagnostic engine, the dynamic revision module, the interaction module and the result assessor. It explains also how they are linked to each other, to the library of models, and to the system's description.

Section 4.4 begins with a description of the overall strategy that forms MVDS. It also recalls briefly the algorithm of the diagnostic engine, that follows GDE's method. Then, it describes and explains the algorithms for the dynamic revisions, interaction module and result assessment, that are the innovative pieces of implementation in MVDS.

The conclusion for this chapter is found in section 4.5. The full program for MVDS is not present in this chapter, but can be found in Appendix A.

4.2 The objects for modelling the system

The different modelling objects involved in the implementation of MVDS fall into three categories. The first category contains the objects used for describing the models of components, i.e. in the library of models, the second contains the objects used for describing the system under consideration, and the third category contains the objects used for representing measurements and predictions.

4.2.1 For describing the components' models

The nature of a component is identified by a name. The nature can be, for example, 'radiator', 'water pump', or 'water softener'. A component is modelled through two types of declarations: the identification of one of its variables, and the declaration of a model associated with this variable. Consider for example a radiator. One variable that can be considered is the water temperature at the outlet port. This can be declared by the following statement:

Variable(radiator, temperature, out).

A model that allows predicting a value for this variable is declared in the following manner:

model(radiator, temp, out, ~<, in).

This means that the variable (radiator, temperature, out) is slightly smaller than the variable (radiator, temp, in), where 'slightly smaller than' is the qualitative operator explained in section 3.4.1.1. Prior to the above declaration of this model, the variable (radiator, temp, in) needs to be declared, as the variable (radiator, temp, out) has already been declared.

In the case of a component with more than two ports, the principle is the same. Names for the ports only need to be different than just 'in' and 'out'. For example, below is a model given for the water temperature in a hot water vessel with four ports:

variable(vessel, temp, in1).
variable(vessel, temp, out1).

variable(vessel, temp, in2).

variable(vessel, temp, out2).

model(vessel, temp, out1, <<, in1).

model(vessel, temp, out2, >>, in2).

Before using this library of models, the physical system under consideration needs to be described by linking the components in its structure with the components' models in this library. This is described in the next section.

4.2.2 For describing the system under consideration

A physical system is described as a network of components having one or more ports. Therefore, a physical system is described in MVDS by listing the components present, how they are linked to each other, and what entities are relevant in the system. Components are declared in the following manner:

Component(r1, radiator),

This declaration means that there is a component referred to as 'r1', and that this component is a radiator.

The relevant entities in the system's behaviour are declared in the following manner:

Relevant_entities (temp, flow, pressure).

The organisation of the system is declared by assigning a location name to components' ports. For example, location(p1, r1, in) means that the port 'in' of the component 'r1' is given the location name 'p1' (as in 'point 1'). If another location declaration is location(p1, r2, out), then the same location point p1 is assigned to the 'out' port of a component 'r2', and therefore the components r1 and r2 are linked, in the manner that the 'out' port of component r2 is linked to the 'in' port of component r1.

Describing the system under consideration also means describing the causal knowledge available about this system. The local and global influences therefore need to be listed. For example, a local influence between the entities E1 and E2, within a component C is declared in this way:

Local_infl (C, E1, E2).

A global influence between two entities E3 and E4 is declared as:

Global_infl (E3, E4).

In order to illustrate the description of a system in MVDS, consider again the air-conditioning example, represented in figure 4.1 below, and its declaration in MVDS.

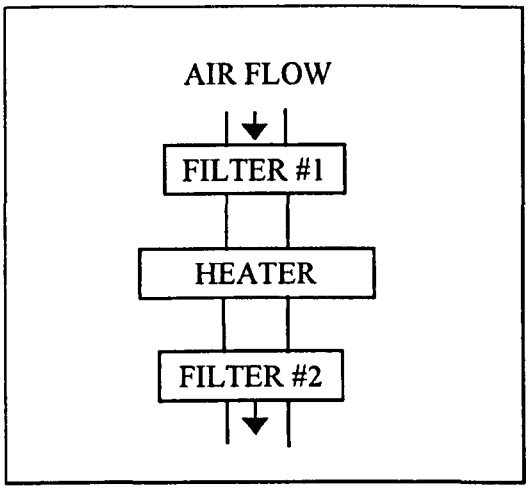


Figure 4.1 - Recall of the 3-piece air-conditioning system

The types of components 'air-heater' and 'air-filter' are assumed to have been declared in the library of models, with the basic two ports 'in' and 'out'. The components' identification can be:

Component(f1, air-filter).

Component(h, air-heater).

Component(f2, air-filter).

Then the system's organisation can be declared:

Location(p1, f1, in).

Location(p2, f1, out).

Location(p2, h, in).

Location(p3, h, out).

Location(p3, f2, in).

Location(p4, f2, out).

Declared in the above manner, the air-conditioning system is viewed by MVDS with the information represented in figure 4.2 below.

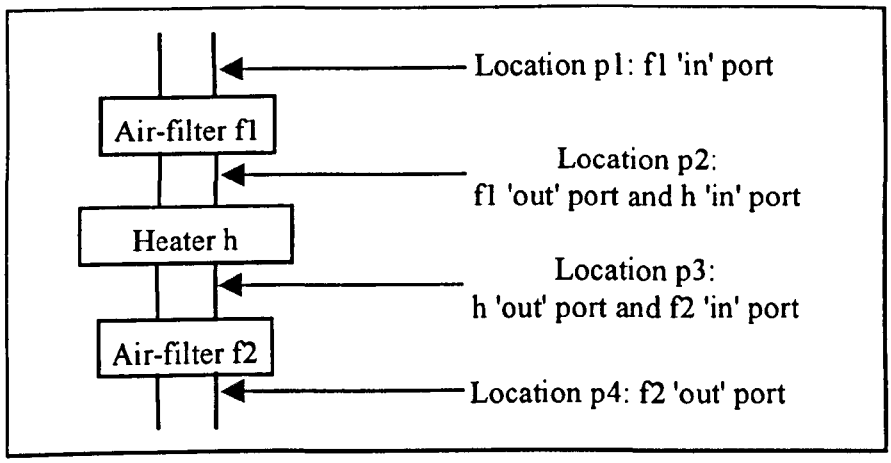


Figure 4.2 – The air-conditioning system as seen by MVDS

The causal knowledge needs also to be declared. Below is an instance of causal knowledge that could be declared for the air-conditioning example:

local_infl(h, dust, temp).

local_infl(h, flow, water).

global_infl(dust, flow).

The physical system is represented for MVDS through its components, links, and causal influences. From this representation, reasoning can be performed. However, undertaking this reasoning requires declaring measurements and representing predictions. These objects are described in the next section.

4.2.3 For declaring measurements and representing predictions

In MVDS' implementation, measurements and predictions are the type of objects: they are quantities. An object *quantity* is formatted in the following way:

Quantity(value, entity, location, environment), where

- *Value* is the qualitative value of the quantity,
- *Entity* is the physical entity of concern,
- *Location* is the physical location of the quantity in the system,
- *Environment* is the set of assumed non-faulty components associated with the quantity. If the quantity is a predicted quantity, then this set contains the components the models of which have

been used for making the prediction. If the quantity is a measured quantity, then this set is empty because the measurement has not been obtained through the use of any component's model.

4.3 The architecture

The architecture of MVDS contains four parts: the interaction module, the diagnostic engine, the dynamic revision module, and the result assessor. The way these parts are linked together is represented by the graph in figure 4.3 below.

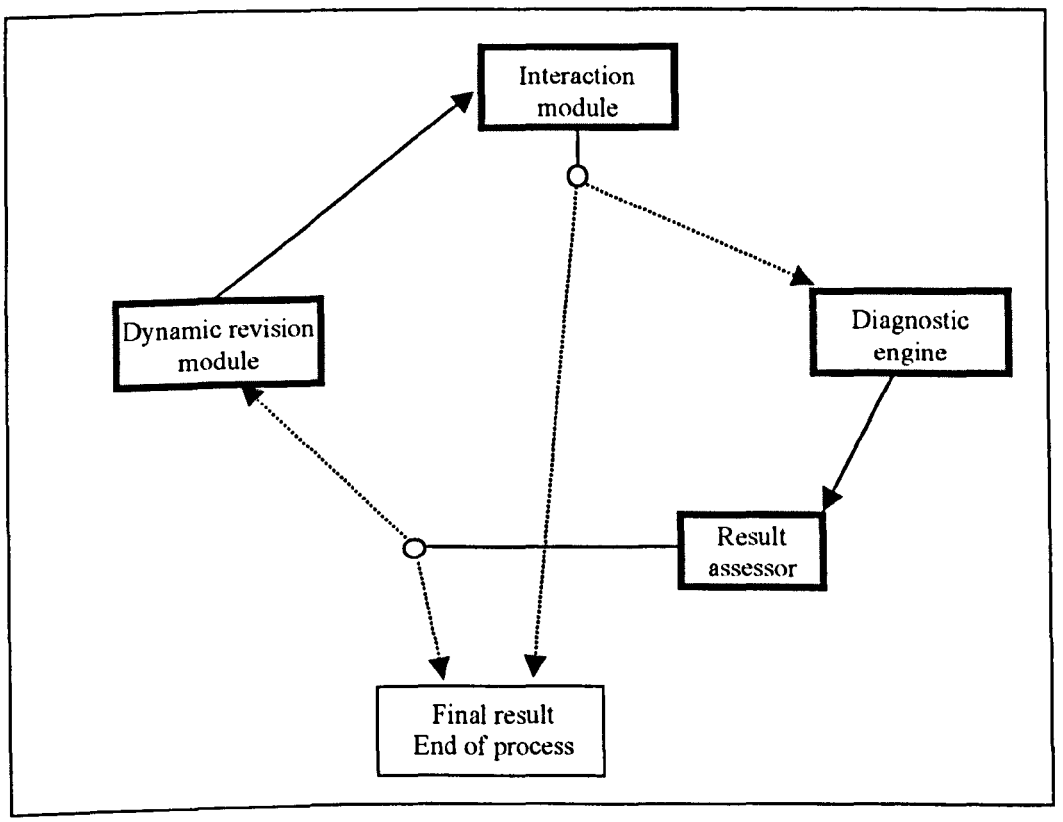


Figure 4.3 - MVDS' conceptual architecture

The four architectural parts are represented by the thick-lined squares, whereas the process termination state is the thin-lined square. A circle followed by two dotted lines represents a fork where the process will follow only one out of the two dotted lines.

4.3.1 Interaction module

The interaction module manages the input of data or knowledge by the operator along the process. It fulfils two main functions: the input of measurements and the input of shallow knowledge.

4.3.1.1 The input of measurements

Before an entity is investigated by the diagnostic engine, measurements are required about this entity. The interaction module prompts the operator to input these measurements. A measurement is a quantity, and must therefore follow the format of a quantity given in section 4.2.3. However, a measurement is a special type of quantity, in that no model has been used in order to obtain it. Thus the *environment* field of a measurement is always an empty set.

Because the set of entities under focus is growing during the diagnostic process, the input of measurements is requested every time another entity is put under focus. Therefore, MVDS prompts the operator to input some measurements about a certain entity every time a dynamic revision happens and before the following diagnosis session.

4.3.1.2 The input of shallow knowledge

In section 3.4.1.4, it was explained how MVDS provides the opportunity for co-operation with the operator. As mentioned, in the current implementation of MVDS, only one co-operative intervention is enabled. It consists of the possibility for the operator to stop the ongoing process if he/she believes that no further dynamic revisions and diagnosis sessions are necessary.

For this purpose, a pause occurs in between each dynamic revision and diagnosis session. There are two reasons for choosing to pause at this stage of the process:

- Firstly, starting another diagnosis session means acquiring beforehand a set of measurements related to the entities to be investigated. The acquisition of this set of measurements is a costly part of the process. Therefore, if the process is going to undertake another diagnosis session, then it is important that no work is required to acquire any useless measurements.

- Secondly, the result of the dynamic revision, *i.e.* which entities are next to be investigated by MVDS, is a useful piece of information that can guide the operator in his/her decision-making task. This is why the pause is located after the dynamic revision has occurred. Ensuring that the operator has all the information that MVDS can give him/her before a co-operative action is taken is important for the quality of the co-operative work. An example of the result of the latest dynamic revision being used for decision-taking would be when the next entity to be investigated is known by the operator to be fault-free. Thus, this entity does not need to be investigated and the process can be stopped while still ensuring that the diagnosis will be entity-complete.

4.3.2 The diagnostic engine

The diagnostic engine follows the interaction module on the organisation graph above. It receives from the interaction module the 'green light' to proceed, after the decision to do so has been made by the operator, as explained in the previous section. The diagnostic engine also receives from the interaction module the set of measurements that is to be used as the base for the diagnosis session. The engine also uses information from the model library and uses the system's description, to allow predictions to be made.

The output of the diagnostic engine consists of conflict sets and candidates, that are transmitted directly to the result assessor.

4.3.3 The result assessor

From the diagnostic engine, the result assessor receives the updated collection of conflict sets and the updated minimal candidates. Its role is to assess whether or not these results need refinement, and if so, whether or not this refinement is possible. If refinement is needed and is possible, then the result assessor triggers the dynamic revision module. Otherwise, the process is ended, and the current candidates are returned as the final diagnosis. The algorithm for the result assessor is described in section 4.4.3.

4.3.4 The dynamic revision module

When the result assessor starts the dynamic revision module, this initiates another loop containing some model reasoning and diagnostic reasoning. The role of the dynamic revision module is to perform the model reasoning and transmit the updated set of entities under focus to the interaction module described in section 4.3.1. The module accesses the physical system's representation. The algorithm for the dynamic revision module is given in section 4.4.4.

4.4 The algorithm

The previous sections have presented the various modules involved in the functioning of MVDS. This section focuses on the algorithmic aspect. Firstly, a description of the general algorithm of MVDS is given. It describes and justifies the overall diagnostic process. The algorithms for a diagnosis session, a result assessment and a dynamic revision, are described respectively in the second, third, and fourth sections.

4.4.1 Overall strategy

The overall organisation of the fault-diagnosis strategy in MVDS consists of a preliminary action and a main body which is a series of loops. However, certain conditions are tested during a loop, that can result in the loop being terminated. This termination can only happen once, for it leads to the termination of the whole process.

Figure 4.4 below represents the flow of control in the algorithm of MVDS. The rectangular boxes represent actions, or groups of actions. The ellipses represent only evaluations, or questions. What happens after one of these evaluations depends on its output.

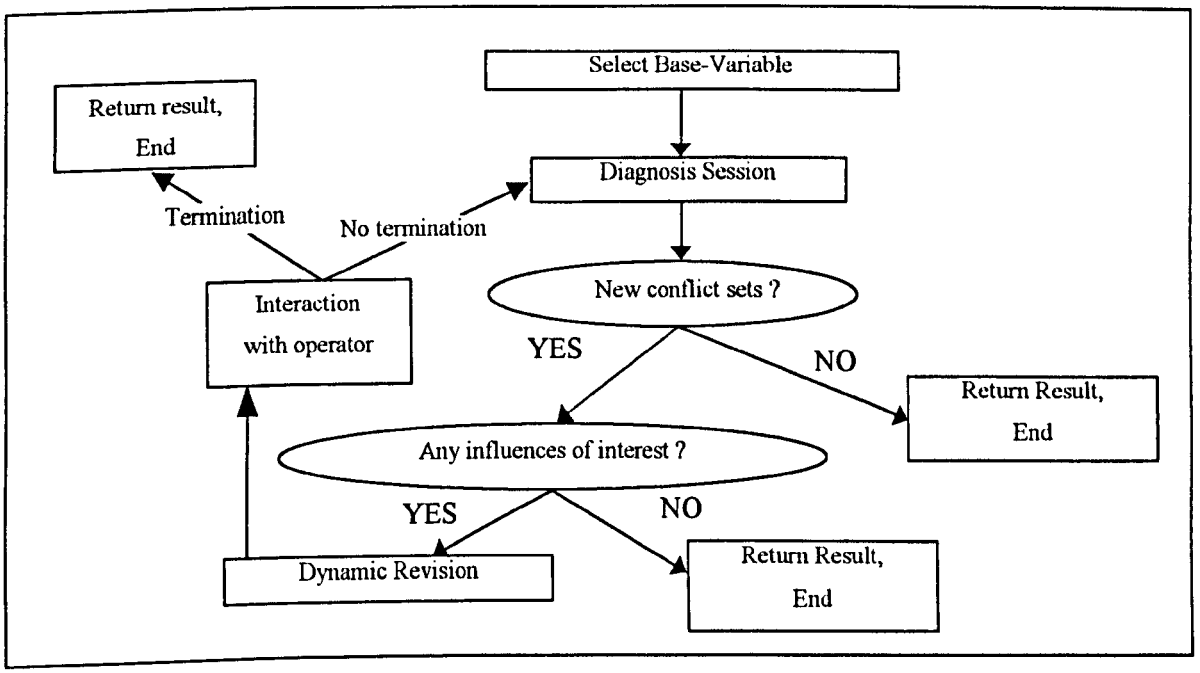


Figure 4.4 - General algorithm for MVDS

Below is the outline of the corresponding algorithm. The numbers have no significance in the algorithm. They are only used to facilitate the explanations which follow.

```
1  Identification of the base-entity.
2  Start of loop.
3          Perform diagnosis session (GDE method),
4          If no new conflict sets, then (return current candidates, end),
5          Else    Look for influences linked to the new conflict sets,
6                  If no linked influence, then (return current candidates, end),
7                  Else    Perform interaction with operator,
8                          If termination, then (return current candidates, end),
9                          Else    Perform dynamic revision,
10                                 Go to start of loop.
```

Preliminary step: before the start of the loops (line 1)

When starting the diagnosis task there is no existing intermediate diagnosis result, but an entity still must be chosen to be under focus in order to run the first diagnosis session. This entity is referred to as the *base entity*. Since the origin of the task is an evidence of a malfunction revealed by an abnormal quantity, the entity corresponding to this quantity is set as the base entity. It is not chosen by the system, since it is known by the operator. It is therefore given by the user as an input to the system.

Main body: the loops (lines 2 to 10)

The algorithm then enters a succession of loops, containing a diagnostic reasoning part (line 3), a result assessment part (lines 4 to 6), an interaction part (lines 7 and 8), and a model reasoning part (line 9). The loops start with a diagnostic reasoning part:

Diagnostic reasoning part:

A diagnosis session is run with the current set of entities under focus, according to the algorithm described in section 4.4.2. The output information contains the updated conflict sets and the previous conflict sets (in the format described earlier), and the minimal candidates.

Result assessment:

The results from the diagnosis session are examined in order to assess the need for further investigation. The algorithm followed for this assessment is detailed in section 4.4.3. If the result is negative, then the process stops and the current candidates are returned as the final result. If the assessment is positive, then a dynamic revision follows.

Dynamic revision:

The role of the dynamic revision is to update the system's representation to take into account the whole set of new conflict sets. The detailed algorithm is given in section 4.4.4. When the dynamic revision is finished, the set of entities under focus has been updated. The process then enters the interaction module.

Interaction with the operator:

As explained in section 4.3.1, two tasks are fulfilled by this module: the input of measurements and the input of shallow knowledge. However, in the current implementation, the latter is restricted to asking the operator whether the process can go on or whether he/she believes that the process can terminate. The flow of control with this restriction is represented in figure 4.5 below. The print messages in this figure are only an indication of the content. The actual content is more complete, for example a message includes a recall of the format in which the measurements should be entered. In the case where the process is not terminated, the input of the list of measurements leads the process to

loop back, thus starting another diagnosis session, with the new set of entities under focus and the corresponding new set of measurements.

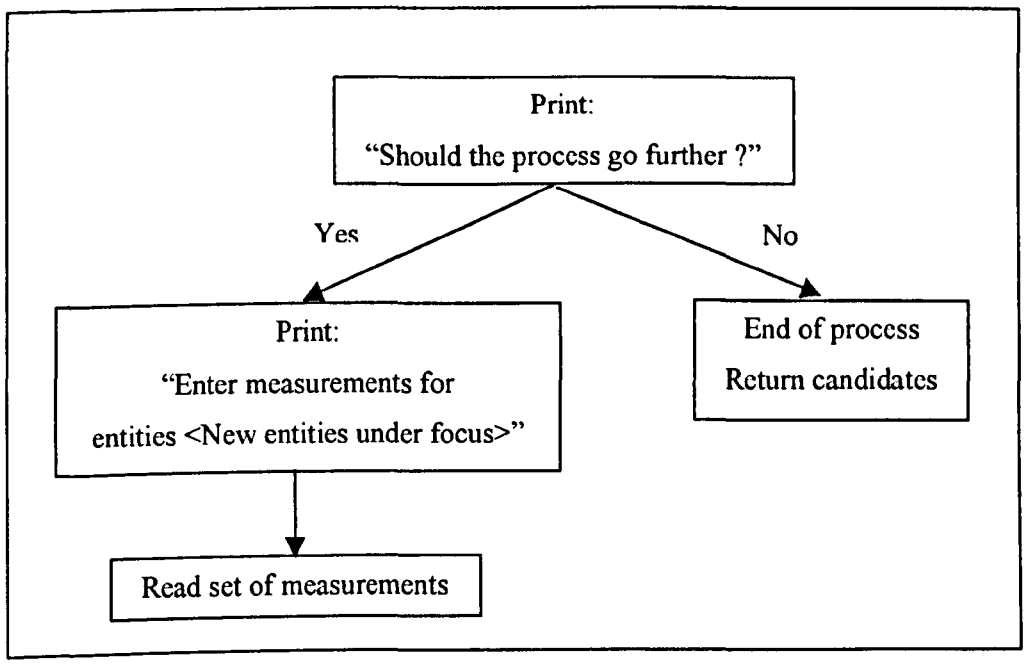


Figure 4.5 – Interaction with the operator

This section has described the overall algorithm, explaining how the process navigates between the different modules of MVDS. In the next two sections, more detailed algorithms are given for the diagnosis session and the dynamic revision.

4.4.2 Diagnosis session

The algorithm for a diagnosis session following the GDE method [de Kleer & Williams, 1987] is recalled below.

Step 1: The measured quantities are propagated through the system's representation, in order to obtain the set of predicted quantities. Each quantity is associated with an environment, which is the list of the components whose models have been used to predict the quantity.

Step 2: The consistency between the measured and predicted quantities, or in between predicted quantities, is checked, resulting in the listing of symptoms. A symptom is an inconsistency

between two predicted quantities or one predicted quantity and a measured quantity. The union of the two contradictory quantities' environments is then a conflict set. At the end of step 2, the output is a list of conflict sets.

Step 3: The set of conflict sets is used to construct the minimal conflict sets. This is obtained by removing each conflict set that is the superset of another or has the same elements than another.

Step 4: The minimal candidates are then constructed from the minimal conflicts sets. First, the candidates are constructed from the previous candidates at the end of the previous diagnosis session and from the new conflict sets. To do so, every candidate that intersects with all the new conflict sets is kept in the new candidates. From each candidate that is not kept, new candidates are constructed by adding to it one component from the new conflict set with which there was no overlap.

An example of this process is given below.

Old candidates: {C1, C2}, {C1, C3}.
New minimal conflict set: {C1, C5, C7} and {C3, C10} (end of step 3).
Construction of the new candidates (step 4):

The candidate {C1, C2} overlaps with the first conflict set but not with the second one. Thus it must be replaced with the candidates obtained by adding one component from the non-overlapping conflict set. These new candidates are {C1, C2, C3} and {C1, C2, C10}.

The candidate {C1, C3} intersects with all the new conflict sets, and thus is kept in the updated set of candidates.

The new set of candidates is therefore {C1, C2, C3}, {C1, C2, C10}, and {C1, C3}.

After the set of candidates is updated, the minimal candidates can be obtained, through the same process of elimination of supersets and sets of identical contents. In the example above, the candidate {C1, C2, C3} is not minimal because it is a superset of the other candidate {C1, C3}. Thus the updated set of minimal candidates is {C1, C2, C10} and {C1, C3}.

The construction of the minimal candidates is the final action in a diagnosis session. The next important subpart of the algorithm of MVDS is the result assessment, described in the next section.

4.4.3 Result assessment

The output of the latest diagnosis session is examined in order to assess how satisfactory it is. This assessment is undertaken through the two evaluations described below.

Evaluation 1:

The output of this evaluation is positive if the latest diagnosis session has revealed new conflict sets, negative otherwise. In the latter case, the process is terminated and the current candidates are returned as the final diagnosis. In the case of a positive evaluation, the process goes on to a second evaluation.

Evaluation 2:

This evaluation is entered with the set of the new conflict sets. The output is positive if these new conflict sets are related to any influence of interest, as explained below. Otherwise, the output is negative, and, as in Evaluation 1, the process is then terminated and the current candidates are returned as the final diagnosis. A conflict set is said to be related to an influence of interest in two cases. First, it can be related to a global influence of interest. This is the case when a new conflict set concerns an entity that is involved in a global influence with an entity that is not under focus. Second, it can be related to a local influence of interest. This happens when a new conflict set originates from a symptom concerned with a quantity that is involved in a local influence with a quantity whose entity is not under focus. Thus, if any influence of interest is found, Evaluation 2 is successful, and this means that the result assessment as a whole returns a positive answer to the question of whether or not the process goes on for further refinements. In this case, a dynamic revision is the next action undertaken by MVDS.

4.4.4 Dynamic revision

A dynamic revision is started after new conflict sets have been revealed and after it has been assessed that they were linked with influences of interest, in the sense explained in section 4.4.1.

The task of a dynamic revision consists of putting under focus, for every entity E1 involved in a newly discovered conflict, the entities E2 such that E1 influences E2 or vice-versa. This means every entity that is globally influenced by or globally influences E1, and every entity which is locally influenced

by or locally influences E1 within a component involved in a conflict. The method followed in a dynamic revision is given below.

Step 1: From the list of new symptoms and new conflict sets, the local influences of interest are identified.

Step 2: Identify the set S1 of entities linked to these local influences.

Step 3: From list of entities involved in new symptoms, identify global influences of interest.

Step 4: Identify the set S2 of entities linked to these global influences.

Step 5: The union of S1 and S2 is the set of entities to be put under focus.

The set of entities to be newly put under focus is returned as the result of the dynamic revision, and this set is submitted to the attention of the operator before the next diagnosis session is started. This co-operation with the operator happens within the interaction module described in section 4.3.1.

4.5 Conclusion

The detail of the implementation and algorithm of MVDS has been examined in this chapter. The conceptual architecture of MVDS is organised around six modules, namely the diagnostic engine, the result assessor, the dynamic revision module, the interaction module, the physical system representation, and the library of models. The overall diagnosis strategy has been explained, by describing how these modules fit together. The individual algorithms or structures of each of these modules have also been described.

This implementation and these algorithms respect the design constraints listed in Chapter 3.

- The qualitative algebra described in section 3.4.1.1 is used for the library of models, as explained in section 4.2.1.
- The algorithm aims to keep the amount of measuring and computation to a minimum, as required in section 3.4.1.2 for coping with complexity.
- As required in section 3.4.1.3, the production of a useful trace is made possible by the use of the GDE method for a diagnosis session.

- An interaction module is included in MVDS, so that co-operation with the operator is conducted, as required in section 3.4.1.4.
- The library of models, implemented with the objects described in section 4.2.1, is structured in independent models of the various entities involved. In that way, it fulfils the requirement expressed in section 3.4.2.1 about the nature of the models.
- As required in section 3.4.2.2, causal knowledge is used. It is declared in the system's representation, and utilised in the dynamic revision module and in the result assessor. As required, it is only used to improve the efficiency of the process, but not in the diagnosis sessions, in order to keep the robustness of a deep-knowledge based process.

The efficiency of this implementation and algorithms is tested in the next chapter, since it describes the use of MVDS on a small example and on a larger case-study. Instances of outputs produced by MVDS are included, in order to exhibit clearly the functioning of the system described in this chapter.

Chapter 5

Application of MVDS: example and case-study

5.1 Introduction

This chapter is concerned with applying MVDS. Diagnostic processes are undertaken with MVDS, using a set of available entities including all the relevant entities of the system at hand, or using a set of available entities reduced to a single entity, in order to represent fault-diagnosis systems using a restricted fixed set of entities. The influence of the sets of measurements used for the diagnostic processes on the possible comparisons is the issue of concern in section 5.2.

In section 5.3, the air-conditioning example is used. Although too simple to reflect seriously the efficiency of MVDS, it has the advantage of offering a clear insight into the functioning of MVDS. It is thus an appropriate example with which to start.

In the next sections, the degree of complexity is increased, as a larger and more complex case-study is used. The system is a water distribution and heating system, described in section 5.4, that is similar to many common domestic installations. In section 5.5, two failure scenarios are set-up, and the diagnosis results obtained by MVDS for each of them are reported. The analysis of these results is the object of chapter 6.

5.2 Measurement sets and performance comparisons

The issue of fixing a set of measurements for a case-study is problematic. Since the set of measurements used for the diagnosis task influences its result, comparisons between different systems' performances must be undertaken with this influence in mind.

The first obvious consequence is that the various systems to be compared must use the same set of measurements. This is because an inefficient diagnosis system using a large set of measurements can obtain a better result than an efficient diagnosis system using a small set of measurements. Running comparative tests with different sets of measurements used for the different diagnosis systems can therefore lead to false conclusions.

The second consequence is more complicated and is related to possible characteristics of a measurement set. For example, a set of measurements about an entity can be small or large (in relation to the size of the system) but can also be concentrated or widely distributed, *i.e.* with most of the measurement points located within a small area of the system or evenly spread across the whole system. Consider then the sets of measurements for the different entities that are investigated. These sets can all be of fairly similar sizes, or some might be large and others might be small. This latter case, where some entities have large sets of measurements and some other entities small sets, is probable in physical systems. This is because some entities are easily measurable, *e.g.* the temperature, but some other entities are less so, *e.g.* pressures.

A way of addressing this problem is to perform a series of comparisons, where the sets of measurements have varying characteristics, as mentioned above. If a diagnosis system obtains better results over the whole range of comparisons, then it is to some extent safe to conclude that this diagnosis system is a better system. Even if no such dominance is noticed, conclusions can still be drawn regarding some characteristics of the diagnosis systems. For example, if a system never performs greatly but reaches a steady level of performance across the range of tests, then the system is probably a robust system. Other systems might obtain some poor results and some excellent results, and can therefore be highly useful systems for diagnosis problems where a certain characteristic of the set of measurements is present.

In view of these considerations, the large case-study in this thesis is investigated with three types of sets of measurements, in order to ensure that the evaluation of the advantages and drawbacks of MVDS is fair. The three types of sets are:

- Ideal: each entity investigated is measured at the ports of the components that are faulty with respect to this entity. No other measurements are made.
- Rich: the measurements are more numerous than in the ideal set, and they are close to the faulty components' ports, but not necessarily as perfectly located as in the ideal set.
- Poor: the number of measurements is the same as in an ideal set, but these measurements are not located at the appropriate ports.

Using these different type of sets of measurements was not the approach taken with the small example described in section 5.3. This is because this air-conditioning example is a motivating example whose most important advantage is simplicity. Only one set of measurements was therefore used for this small example, and this set was chosen to demonstrate, in the most striking possible way, the potential advantage of using MVDS.

5.3 A small but motivating example

Before using MVDS on a larger case-study, the small air-conditioning example from section 3.2.1 is used to illustrate the diagnostic processes undertaken by MVDS. For ease of reading, the example is completely presented again in the following section, together with the fault-scenario. The library of models that is necessary in order to run MVDS on this example is described in section 5.3.2. The declaration of the air-conditioning system in the format of MVDS is explained in section 5.3.3, and a justification of the measurement that will be used is given in section 5.3.4. The diagnostic performance of MVDS is described in section 5.3.5, accompanied by the output which is produced. In this same section, the issue of which diagnosis is considered to be satisfactory is addressed. In this respect, the performance of MVDS is compared with the performance of approaches that use a fixed set of entities under focus.

5.3.1 The 3-piece air-conditioning system and the fault-scenario

Consider an air conditioning system with a heater and two filters retaining water and dust. On the schematic in figure 5.1, the pointers P1 to P4 designate the locations where it is possible to make measurements. The variables considered are the flow of air, its temperature, the density of dust and the density of water in this flow of air.

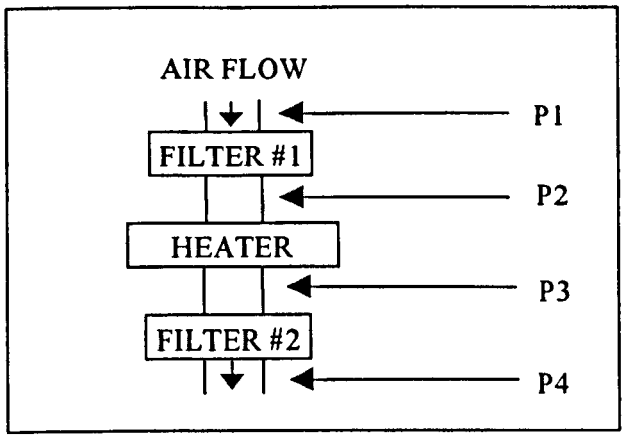


Figure 5.1 – Recall of the 3-piece air-conditioning system

The filter #1 is present in order to avoid too much dust and humidity getting into the heater. The heater can be just a simple electric resistor. The filter #2 is present to remove all the dust and humidity that was not removed by filter #1 or got in the air on the way through the heater. The air-flow coming out at location P4 should therefore be hot and free of dust and humidity.

The fault scenario is the following. Filter #1 at the air entry is damaged, and as a consequence does not filter enough dust. Because of filter #2 present at the air outlet, the outgoing flow of air is not especially overcharged with dust or water. Thus no fault is noticeable by the user immediately after the failure of filter #1. The system therefore remains functioning, and the heater receives a flow of dust-loaded air. This damages it and after a while it becomes inefficient. The consequence, which is an important decrease of the temperature of the outflow of air, is noticed by the user, and the system is stopped for repair. Although simple, this scenario has led to a complex multiple-fault situation, where two components are faulty, with respect to two distinct entities.

5.3.2 The library of models

The library of models needs to contain the models for an air-filter and an air-heater. They are based on the following principles:

- Between the input and output ports, an air-filter does not affect significantly the temperature of the air nor its flow, but considerably reduces the content of dust and humidity in the air.
- Between the input and output ports, an air-heater increases the air temperature considerably, and also slightly increases its dust and humidity content, as a result of the condensation and material ageing process. It does not affect significantly the flow.

The models below are the simple models applied by MVDS. The input and output ports of the components are named respectively <in> and <out>.

Model of an air-filter:

```
variable(airfilter, flow, in).
variable(airfilter, flow, out).        model(airfilter, flow, in, ==, out).
variable(airfilter, dust, in).
variable(airfilter, dust, out).        model(airfilter, dust, in, >>, out).
variable(airfilter, water, in).
variable(airfilter, water, out).       model(airfilter, water, in, >>, out).
variable(airfilter, temp, in).
variable(airfilter, temp, out).        model(airfilter, temp, in, ==, out).
```

Model of an air-heater:

```
variable(airheater, flow, in).
variable(airheater, flow, out).        model(airheater, flow, in, ==, out).
variable(airheater, dust, in).
variable(airheater, dust, out).        model(airheater, dust, in, ~<, out).
variable(airheater, water, in).
variable(airheater, water, out).       model(airheater, water, in, ~<, out).
variable(airheater, temp, in).
variable(airheater, temp, out).        model(airheater, temp, in, <<, out).
```

5.3.3 Declaring the system in MVDS

This section explains how the 3-piece air-conditioning system is declared in the implementation of MVDS. It follows the design explained in section 4.2.2.

Entities:	Relevant_entities(temp, dust, water, flow).
Components:	component(f1, airfilter).
	component(f2, airfilter).
	component(h, airheater).
Structure:	location(p1, f1, in).
	location(p2, f1, out).
	location(p2, h, in).
	location(p3, h, out).
	location(p3, f2, in).
	location(p4, f2, out).
Causal knowledge:	local_infl(h, dust, temp).
	local_infl(h, flow, water).
	global_infl(dust, flow).

As introduced in section 3.4.2.2, this causal knowledge can be represented graphically in a graph of causal dependencies, as given in figure 5.2 below.

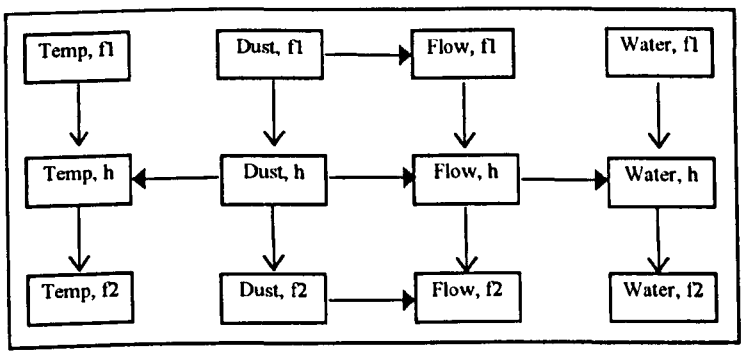


Figure 5.2 - Graph of causal dependencies for the air-conditioning example

5.3.4 The measurements

The set of measurements, entered by the user when prompted by MVDS, is chosen arbitrarily for its ability to demonstrate clearly the potential advantage of MVDS, as justified in section 5.2. This set of measurements is:

- For the temperature: Low at location p2 and Medium at location p3. This represents the fact that the heater is performing poorly, since Medium ~> Low, whereas the model for an air heater is Temp out >> Temp in. Following the format for a measurement or a prediction in MVDS, as described in section 4.2.3, this is entered as:

  Value(low, temp, p2, [])

  Value(medium, temp, p3, [])

- For the dust: High at location p1 and Medium at location p2. This represents the fact that the filter #1 is not filtering enough dust, since Medium ~< High, whereas the model for an air-filter is Dust out << Dust in. This is entered as:

  Value(high, dust, p1, [])

  Value(medium, dust, p2, [])

- For the flow: High at locations p1 and p4. This represents the fact that no obstacle to the flow is present. This is entered as:

  Value(high, flow, p1, [])

  Value(high, flow, p4, [])

5.3.5 The diagnostic performances: fixed sets under focus and MVDS

The declarations above, of the models and of the structure of the physical system at hand, allow MVDS to undertake a diagnostic task. Three outputs of diagnosis tasks performed by MVDS are printed in this section. In order to compare the performance of MVDS against other systems, the different results obtained are examined, starting with a satisfactory diagnosis.

5.3.5.1 Defining the satisfactory diagnosis

The usual characterisation of the set of candidates by the set of minimal candidates [de Kleer and Williams, 1987] is used here. In that case, it is common practice to consider as the final diagnosis the dominant (in terms of likelihood) minimal candidate. Therefore, in a case where components C1 and C2 are the faulty components, reaching the minimal candidate {C1} is a correct result, but which would actually lead to wrong or incomplete actions. Indeed, {C1} being taken as the final result, the component C1 would be the only component subject to actions as a consequence of the diagnosis task. For instance, in the air-conditioning example, consider the case where the final minimal candidate is {Heater}. By "final" it is meant that no further discrimination is possible. The diagnosis {Heater, Filter #1} is consistent with the minimal candidate {Heater}, but is not more probable than {Heater, Filter #2}, and there is no indication that the situation includes multiple-faults. Therefore, although {Heater} is the minimal candidate, it would probably be considered as the diagnosis. The heater will be either re-conditioned or replaced, and the air-conditioning system will be re-started. However, because the filter #1 has not been replaced, the same fault-scenario is happening straight away again.

In order to avoid this problem, the final minimal candidate needs to be {Heater, Filter #1}. In general terms, a satisfactory result should therefore consist of a minimal candidate matching the set of faulty components. This is what MVDS aims to obtain.

5.3.5.2 The diagnosis by MVDS

Below is the output of the diagnostic performance by MVDS, where, applied to the air-conditioning example, it obtains the satisfactory diagnosis {Heater, Filter #1}. It is followed by the explanation of this process. The numbered marks in the output (in a different bold font) have been inserted for ease of reference in the explanation.

| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(low, temp, p2, []), value(medium, temp, p3, [])].

------ Predictions obtained about entity [temp] -----

value(low,temp,p1,[f1])
value(high,temp,p3,[h])
value(medium,temp,p4,[f2])
value(very_low,temp,p2,[h])
value(high,temp,p4,[h,f2])
value(very_low,temp,p1,[h,f1])

--------- New symptoms and conflict sets ------(Mark 1)

Symptom at loc. p2 : temp = low vs temp = very_low
(resulting conflict set: [h])
Symptom at loc. p3 : temp = medium vs temp = high
(resulting conflict set: [h])
Symptom at loc. p1 : temp = low vs temp = very_low
(resulting conflict set: [h,f1])
Symptom at loc. p4 : temp = medium vs temp = high
(resulting conflict set: [h,f2])

------ Updated minimal conflict sets ------ (Mark 2)

[h]

--------- Updated minimal candidates ----------

[h]

**** Result assessment **** (Mark 3)

> Global influences:
No global influence of interest.
> Local influences:
Entity <dust> put under focus because local influence
between <dust> and <temp> (involved in conflicts),
within component h
*** End of result assessment and start of dynamic revision***

The new entities under focus are [dust] (Mark 4)

*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity dust, with format:
[value(qualitat. value, dust, location, []), value(...), ...].

Measurements |: [value(high, dust, p1, []), value(medium, dust, p2, [])].

(Mark 5)

------- Predictions obtained about entity [dust] -----

value(low,dust,p2,[f1])
value(very_high,dust,p1,[f1])
value(high,dust,p3,[h])
value(medium,dust,p3,[f1,h])
value(low,dust,p4,[h,f2])
value(very_low,dust,p4,[f1,h,f2])

---------- New symptoms and conflict sets ----------

Symptom at loc. p1 : dust = high vs dust = very_high
(resulting conflict set: [f1])
Symptom at loc. p2 : dust = medium vs dust = low
(resulting conflict set: [f1])
Symptom at loc. p3 : dust = high vs dust = medium
(resulting conflict set: [f1,h])
Symptom at loc. p4 : dust = low vs dust = very_low
(resulting conflict set: [f1,h,f2])

------- Updated minimal conflict sets --------

[h]
[f1]

---------- Updated minimal candidates ---- (Mark 6)

[h,f1]

**** Result assessment ****

> Global influences:

```
Entity <flow> put under focus because global influence
between <flow> and <dust> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow]      (Mark 7)
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p1, []), value(high, flow, p4,
[])].

------ Predictions obtained about entity [flow] ----
value(high,flow,p2,[f1])
value(high,flow,p3,[f2])
value(high,flow,p3,[f1,h])
value(high,flow,p2,[f2,h])
value(high,flow,p4,[f1,h,f2])
value(high,flow,p1,[f2,h,f1])

---------- New symptoms and conflict sets ----------
                                                (Mark 8)
------- Updated minimal conflict sets --------
[h]
[f1]

---------- Updated minimal candidates ----------
[h,f1]

**** Result assessment ****
Last diagnosis session revealed no new conflicts => End
                                                (Mark 9)
*** End of result assessment ***
*** End of process ***
```

Explanation of the above diagnosis task:

After the measurements for the entity Temp are entered, the first diagnosis session reveals four symptoms (mark 1), using the six predictions made and the measurements. The resulting minimal conflict set is {h} (mark 2), standing for {Heater}.

The result assessment (mark 3) does not acknowledge any global influence of interest but one local influence of interest. It is the influence between the entities Temp and Dust within the component Heater. This influence is of interest because the entity Dust is not under focus yet and the heater is included in a conflict set affecting the Temp. Therefore the dynamic revision results in the entity Dust being put under focus (mark 4). The interaction module requests the authorisation to go further, and once obtained, asks for the measurements made about Dust to be entered.

Once these measurements have been entered (mark 5), the next diagnosis session makes predictions about the entity Dust, and reveals four symptoms affecting this entity. Considering all the conflict sets discovered by reasoning on Temp and on Dust, the updated minimal conflict sets are then {h} and {f1}, standing for {Heater} and {Filter #1}. The corresponding updated minimal candidate is {Heater, Filter #1} (mark 6).

The result assessment does not point out any local influence of interest but the global influence between Flow and Dust. Because the investigation of Dust has revealed new symptoms, the dynamic revision results in the entity Flow being put under focus (mark 7).
The interaction module receives the authorisation to go further and gets the measurements entered about Flow. However, the next diagnosis session does not reveal any symptoms affecting the entity Flow (mark 8). As a consequence, the minimal candidate {Heater, Filter #1} stays unchanged, and the dynamic revision induces the termination of the process (mark 9).
The final minimal candidate is thus {Heater, Filter #1}, which is the satisfactory result, as claimed in the previous section.

5.3.5.3 Diagnosis obtained with a fixed set of entities under focus

With a traditional consistency-based approach where the set of entities under focus is fixed from the start to the end of the process, the behaviour of the physical system is typically represented either by the behaviour of its main functional entity (*e.g.* the intensity of current in an electrical circuit) or by the entity that revealed the malfunction at the origin of the task.

In the present example, the variable used for the diagnosis task would typically be 'temperature' because it is the entity of which unexpected behaviour is the starting point of the diagnostic task. Using the same measurements as with MVDS, the resulting minimal candidate would be {Heater} (see below the line "Updated minimal candidates"), instead of the satisfactory {Heater, Filter#1}. This is illustrated by the following output, produced by MVDS when the set of entities available is restricted to only {Temperature}. This output is therefore produced by MVDS, but, because of the restricted working conditions, it represents the process followed by a traditional system.

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(low, temp, p2, []), value(medium,
temp, p3, [])].

------ Predictions obtained about entity [temp] -----

value(low,temp,p1,[f1])
value(high,temp,p3,[h])
value(medium,temp,p4,[f2])
value(very_low,temp,p2,[h])
value(high,temp,p4,[h,f2])
value(very_low,temp,p1,[h,f1])

---------- New symptoms and conflict sets ----------

Symptom at loc. p2 : temp = low vs temp = very_low
 (resulting conflict set: [h] )
Symptom at loc. p3 : temp = medium vs temp = high
 (resulting conflict set: [h] )
Symptom at loc. p1 : temp = low vs temp = very_low
 (resulting conflict set: [h,f1] )
Symptom at loc. p4 : temp = medium vs temp = high
```

```
(resulting conflict set: [h,f2] )

------ Updated minimal conflict sets -------
[h]

-------- Updated minimal candidates ----------
[h]

**** Result assessment ****
All available entities are already under focus.
*** End of result assessment ***
*** End of process ***
```

If, for any reason, the set of available entities is fixed to {dust}, the minimal candidate obtained would be {Filter#1} (see below the line "Updated minimal candidates"), still different to the satisfactory result. This is illustrated by the output printed below, produced by MVDS when the set of available entities is restricted to {dust}.

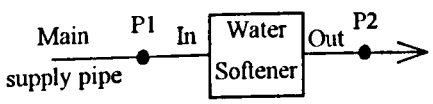
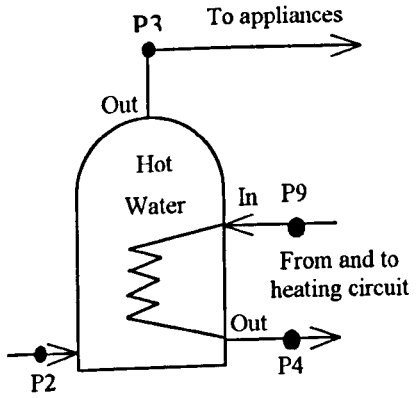
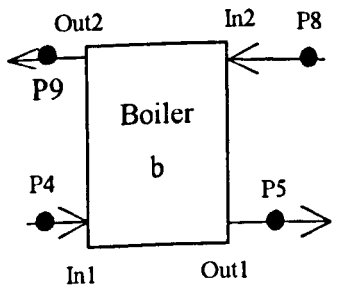
```
| ?- candidates([dust]).

Enter the measurements for entity dust, with format:
[value(qualitat. value, dust, location, []), value(...), ...].

Measurements |: [value(high, dust, p1, []), value(medium,
dust, p2, [])].

------ Predictions obtained about entity [dust] ----

value(low,dust,p2,[f1])
value(very_high,dust,p1,[f1])
value(high,dust,p3,[h])
value(medium,dust,p3,[f1,h])
value(low,dust,p4,[h,f2])
value(very_low,dust,p4,[f1,h,f2])

-------- New symptoms and conflict sets --------
Symptom at loc. p1 : dust = high vs dust = very_high
 (resulting conflict set: [f1] )
Symptom at loc. p2 : dust = medium vs dust = low
 (resulting conflict set: [f1] )
Symptom at loc. p3 : dust = high vs dust = medium
 (resulting conflict set: [f1,h] )
Symptom at loc. p4 : dust = low vs dust = very_low
 (resulting conflict set: [f1,h,f2] )

------ Updated minimal conflict sets -------
[f1]

-------- Updated minimal candidates ----------
[f1]

**** Result assessment ****
All available entities are already under focus.
*** End of result assessment ***
*** End of process ***
```

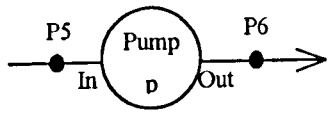
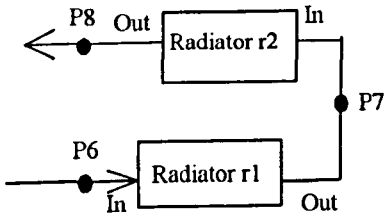
5.4 A larger case–study

The case-study is concerned with a hot-water and heating system, as introduced in a general manner in section 3.3. The particular system used for this case-study contains enough components of different types to represent a usual domestic hot-water and heating system.

5.4.1 Structural description

The system can be separated in two sub-circuits that are, however, not independent to each other. The first one is the hot water circuit, since it fulfils the task of heating water and distributing it to some appliances that are not detailed in this case-study. The second circuit is the heating circuit, since it is ensures the heating and circulation of a closed water flow through, amongst other components, two radiators. Below is the list of the components and their role in the system.

	After the main water supply valve, a water softener ensures that the water in the hot water circuit has a low hardness. This is often necessary for industrial appliances where the formation of limescale has to be reduced, or where water of low hardness is needed (*e.g.* in photographic or chemical processes).
	A hot water vessel makes the link between the hot water and the heating circuit. The water flow from the softener fills this vessel through the inlet port In2, located at its base. A coiled pipe, coming from the heating circuit, is present in the centre of the vessel, entering through port In1 and leaving through port Out1. This pipe conducts the flow of hot water, causing the water in the vessel to heat up by contact. The heated water rises up to the top of the vessel where the outlet Out2 is located, linked to a pipe leading to the appliances.
	The heating circuit contains a boiler b with two inlet pipes In1 and In2, and two outlet pipes, Out1 and Out2. Through In1 and In2, the boiler receives water flows from radiators or from the vessel, to be reheated. The hot water then goes out through Out1 and Out 2 towards the vessel and the radiators again. By doing so, the boiler keeps the water temperature in the heating circuit constantly high.

	Following the Out1 port of the boiler, a pump p is located. This keeps the water in the heating circuit flowing.
	In the heating circuit, after the Out port of the pump, there are two radiators r1 and r2. They ensure the transmission of heat between the water flow and the surrounding air. They are linked in series, and the Out port of r2 is linked to the boiler.

The system is represented on the graph in figure 5.3, where lines represent the pipes in which the water flows, and arrows represent the direction of flow.

5.4.2 Behavioural description

The entities considered in this case-study are the flow, the pressure, the temperature, the air content, and the hardness (or more generally the content of mineral matters) of the water. The components of this hot water and heating system are modelled for MVDS following their basic behaviours. These are the behavioural features that are taken into account when writing the models for MVDS given in section 5.5.1. For each component, the entities that are not named are not significantly changed.

- A water softener severely decreases the water hardness.
- In a hot water vessel, the temperature of the main water volume increases severely, and the temperature of the water in the coiled pipe decreases severely.
- In a boiler, the temperatures of the two flows are severely increased.
- A pump severely increases the pressure.
- A radiator slightly decreases the temperature of the water flow.

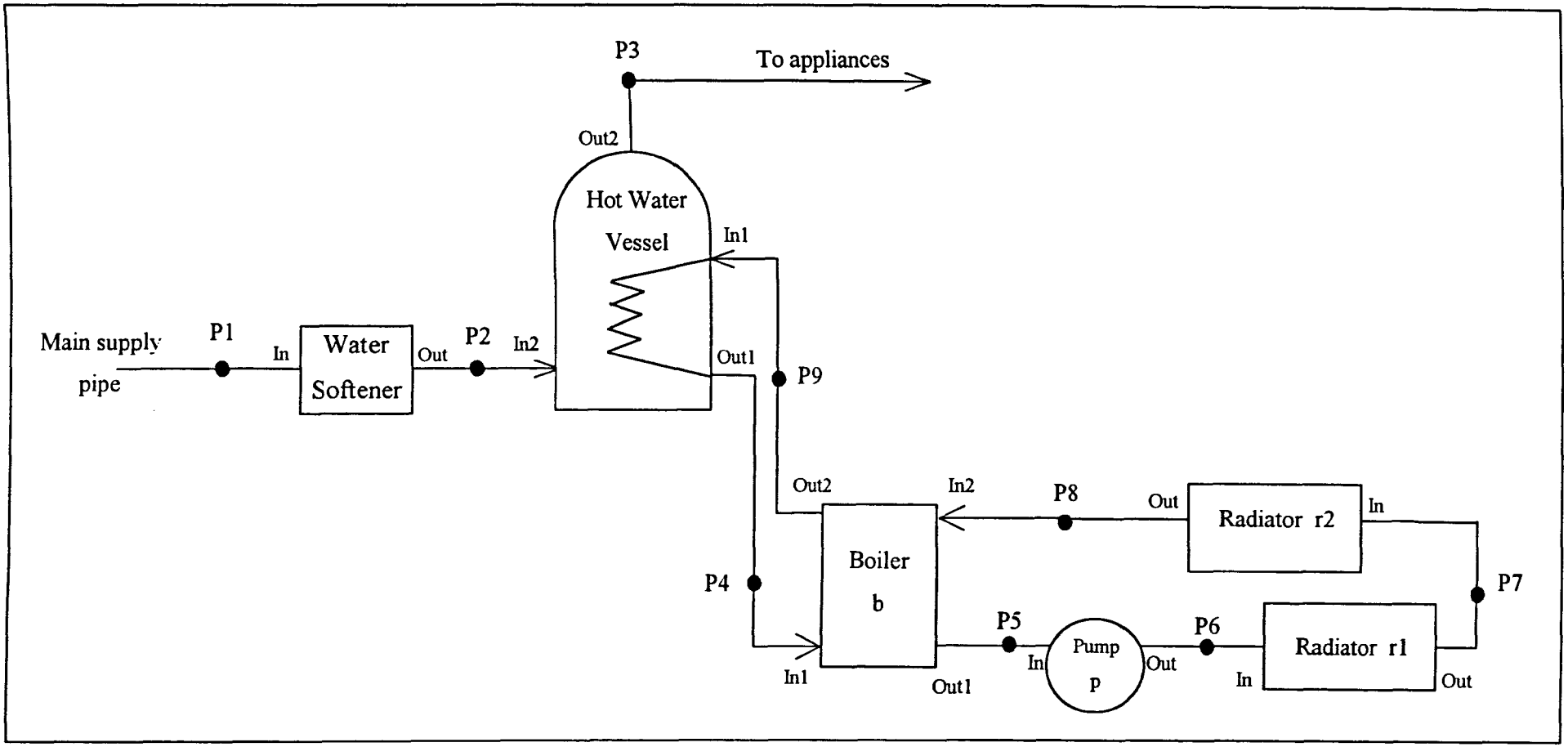


Figure 5.3 - The case-study: a hot-water and heating system

The influences between entities and quantities that are considered are explained below.

- There is a local influence between the flow and the temperature within a radiator. Indeed, if the flow is lower, then the gradient of temperature is greater.
- There is a global influence between hardness and flow, since the concentration of particles carried in hard water can cause the creation of deposits, inducing a slowing down of the flow.
- There is a global influence between pressure and flow, since these two entities are proportional in a water flow.
- There is a local influence between the air content and the temperature in a radiator. Indeed, the formation of air pockets in a radiator, due to an entering flow with a high air content, lowers down the gradient of temperature of the water across the radiator.

The structure and the important behavioural features of the hot water and heating system described in this section need to be declared in MVDS, before running a diagnostic process. This declaration is described in the next section, followed by descriptions of some fault-scenarios and their corresponding diagnosis.

5.5 Using MVDS on the case-study

5.5.1 Declaration of the system's structure

In MVDS, the structure of the hot water and heating system is declared as below, following the format described in section 4.2.2:

```
component(b2, boilerfour).
component(r1, radiator).
component(r2, radiator).
component(p, waterpump).
component(v, vessel).
component(s, softener).

locate(p1, s, in).
locate(p2, s, out).
locate(p2, v, in2).
locate(p3, v, out2).
locate(p4, v, out1).
locate(p4, b2, in1).
locate(p5, b2, out1).
locate(p5, p, in).

locate(p6, p, out).
locate(p6, r1, in).
locate(p7, r1, out).
locate(p7, r2, in).
locate(p8, r2, out).
locate(p8, b2, in2).
locate(p9, b2, out2).
locate(p9, v, in1).
```

The boiler is named "boilerfour" to refer to the model of a boiler with four ports, as opposed to a simpler kind of boiler with two ports.

5.5.2 The library of models

In order to run MVDS on the case-study, the library contains the models for the five types of components included in the system, namely water softeners, 4-port boilers, water pumps, radiators and hot-water vessels. The models declared in the library, following the format described in section 4.2.1, are listed below.

For a water softener with inlet <in> and outlet <out>:

```
variable(softener, flow, in).
variable(softener, flow, out).
model(softener, flow, in, ==, out).

variable(softener, pressure, in).
variable(softener, pressure, out).
model(softener, pressure, in, ==, out).

variable(softener, temp, in).
variable(softener, temp, out).
model(softener, temp, out, ==, in).

variable(softener, hard, in).
variable(softener, hard, out).
model(softener, hard, in, >>, out).

variable(softener, air, in).
variable(softener, air, out).
model(softener, air, in, ==, out).
```

For a water boiler <boilerfour> with inlets <in1>and <in2>, and outlets <out1> and <out2>:

```
variable(boilerfour, flow, in1).
variable(boilerfour, flow, out1).
variable(boilerfour, flow, in2).
variable(boilerfour, flow, out2).
model(boilerfour, flow, in1, ==, out1).
model(boilerfour, flow, in2, ==, out2).

variable(boilerfour, pressure, in1).
variable(boilerfour, pressure, out1).
variable(boilerfour, pressure, in2).

variable(boilerfour, temp, in1).
variable(boilerfour, temp, out1).
variable(boilerfour, temp, in2).
variable(boilerfour, temp, out2).
model(boilerfour, temp, in1,<<, out1).
model(boilerfour, temp, in2,<<, out2).

variable(boilerfour, hard, in1).
variable(boilerfour, hard, out1).
variable(boilerfour, hard, in2).
```

variable(boilerfour, pressure, out2).
model(boilerfour, pressure, in1,==, out1).
model(boilerfour, pressure, in2,==, out2).

variable(boilerfour, hard, out2).
model(boilerfour, hard, in1, ==, out1).
model(boilerfour, hard, in2, ==, out2).

variable(boilerfour, air, in1).
variable(boilerfour, air, out1).
variable(boilerfour, air, in2).
variable(boilerfour, air, out2).
model(boilerfour, air, in1,==, out1).
model(boilerfour, air, in2,==, out2).

For a hot-water vessel <vessel> with inlets <in1>and <in2>, and outlets <out1> and <out2>:
(NB: The ports with suffix <1> relate to the flow from the boiler, and the ports with suffix <2> relate to the flow of water stored.)

variable(vessel, flow, in1).
variable(vessel, flow, out1).
variable(vessel, flow, in2).
variable(vessel, flow, out2).
model(vessel, flow, in1, ==, out1).
model(vessel, flow, in2, ==, out2).

variable(vessel, temp, in1).
variable(vessel, temp, out1).
variable(vessel, temp, in2).
variable(vessel, temp, out2).
model(vessel, temp, in1,>>, out1).
model(vessel, temp, in2,<<, out2).

variable(vessel, pressure, in1).
variable(vessel, pressure, out1).
variable(vessel, pressure, in2).
variable(vessel, pressure, out2).
model(vessel, pressure, in1,==, out1).
model(vessel, pressure, in2,==, out2).

variable(vessel, hard, in1).
variable(vessel, hard, out1).
variable(vessel, hard, in2).
variable(vessel, hard, out2).
model(vessel, hard, in1, ==, out1).
model(vessel, hard, in2, ==, out2).

variable(vessel, air, in1).
variable(vessel, air, out1).
variable(vessel, air, in2).
variable(vessel, air, out2).
model(vessel, air, in1,==, out1).
model(vessel, air, in2,==, out2).

For a water pump <waterpump> with inlet <in> and outlet <out>:

```
variable(waterpump, flow, in).
variable(waterpump, flow, out).
model(waterpump, flow, in, ==, out).

variable(waterpump, pressure, in).
variable(waterpump, pressure, out).
model(waterpump, pressure, in, <<, out).

variable(waterpump, air, in).
variable(waterpump, air, out).
model(waterpump, air, in, ==, out).

variable(waterpump, temp, in).
variable(waterpump, temp, out).
model(waterpump, temp, in, ==, out).

variable(waterpump, hard, in).
variable(waterpump, hard, out).
model(waterpump, hard, in, ==, out).
```

For a radiator with inlet <in> and outlet <out>:

```
variable(radiator, flow, in).
variable(radiator, flow, out).
model(radiator, flow, in, ==, out).

variable(radiator, pressure, in).
variable(radiator, pressure, out).
model(radiator, pressure, in, ==, out).

variable(radiator, air, in).
variable(radiator, air, out).
model(radiator, air, in, ==, out).

variable(radiator, temp, in).
variable(radiator, temp, out).
model(radiator, temp, in, ~>, out).

variable(radiator, hard, in).
variable(radiator, hard, out).
model(radiator, hard, in, ==, out).
```

5.5.3 The causal knowledge

The entities and influences considered in this case-study have been listed in section 5.4.2. Below is their declaration in MVDS, following the format described in section 4.2.2.

relevant_entities([temp, flow, pressure, air, hard]).

local_infl(r1, flow, temp).

local_infl(r2, flow, temp).

local_infl(r1, air, temp).

local_infl(v, flow, temp).

local_infl(r2, air, temp).

global_infl(hard, flow).

global_infl(pressure, flow).

The graph of causal dependencies, corresponding to the causal knowledge above, is presented in figure 5.4 below. The smaller arrows, in the vertical direction, are the influences that exist internally to an entity. These influences are assumed to be always present, and are only represented here for the sake of consistency. The influences between different entities are represented by arrows with triangular and shaded in heads.

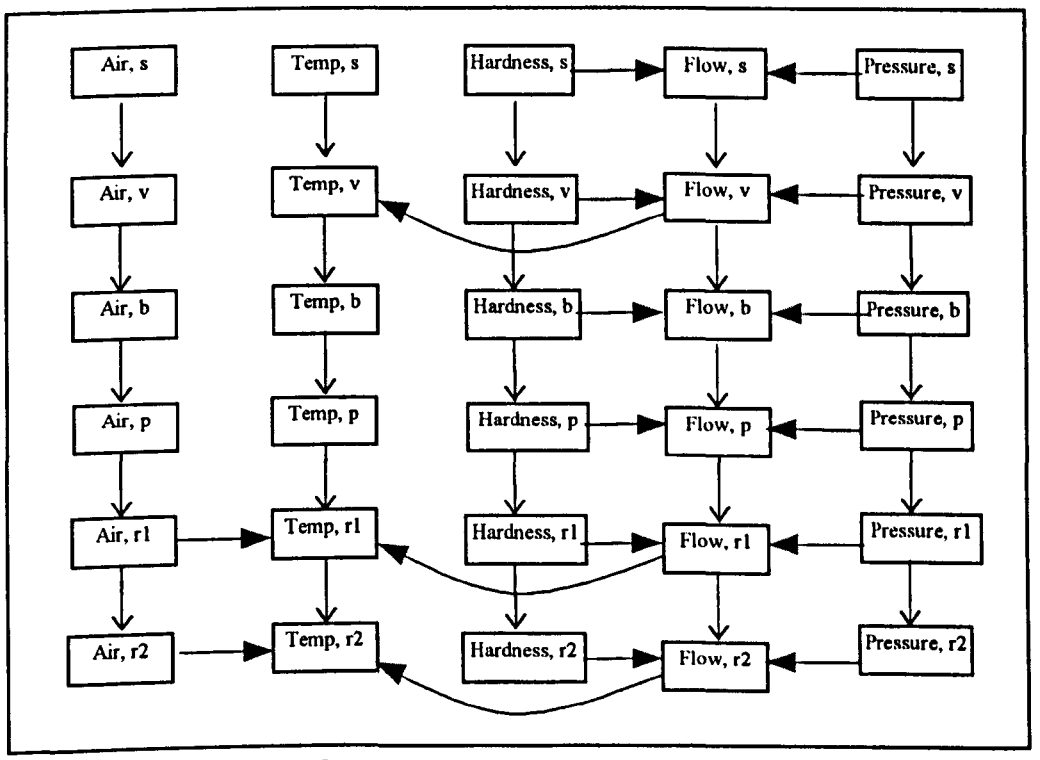


Figure 5.4 - Graph of causal dependencies for the study-case

5.5.4 Scenarios of failure

Scenario 1:

Because of a defective inlet valve that allows a slow air intake in radiator r1, the air content after r1 is increased, and some air pockets have formed in the radiator. However, because the air pockets are regularly evacuated by the mean of regular bleedings of the radiators, the decrease of temperature induced by their presence is not dramatic enough to be noticed. Beside the air pockets, the presence of high quantities of air in the whole circuit has an important consequence on the boiler: the heat and the high air content results in the dissolved mineral particles solidifying into a scale deposit in the boiler. This accumulation of scale deposit induces a decrease of the boiler's efficiency, and therefore a low temperature gain across the boiler. The temperature in the whole system is therefore lower than expected.

The water temperature is therefore the base-entity, since it is the observation of this entity showing an unexpected behaviour that is at the origin of the diagnostic process. The components that need repairing are the radiator r1, in order to stop the air intake, and the boiler b2 that needs to be de-scaled in order to regain its normal efficiency. {r1, b2} is therefore the satisfactory minimal candidate.

Scenario 2:

The original fault is that the seals in the pump p are beginning to leak. This can be part of a normal ageing process. As a consequence, the flow of water at the outlet of the pump is lower than the flow at the inlet. The decrease of the flow is general, and allows a slow formation and deposit of sludge in the radiator r1. The consequence of this sludge accumulation is the decrease of the heat transfer from the radiator to the surrounding air.

The temperature is thus the base-entity again, since the low efficiency of the radiator r1 is the origin of the diagnostic task. The faulty components are the pump p, since it needs to be replaced (or the seals changed), and the radiator r1, since it needs to be replaced or emptied of sludge. {r1, p} is therefore the satisfactory minimal candidate.

5.5.5 Diagnostic processes performed on scenario 1

This section lists the diagnosis results obtained with MVDS or obtained with a fixed set of available entities. MVDS is tested against several sets of measurements having different characteristics. The same sets of measurements are used to diagnose the hot water and heating system using a fixed set of entities, in order to evaluate the benefits of enabling dynamic adaptation of the set of entities under focus.

The sets of measurements which are entered by the user along the process when they are requested are the following:

Ideal set: Temperature: Very high at P6 and very high at P7,
Flow: High at P5 and high at P6,
Air content: Low at P6 and medium at P7,
Hardness: Low at P8 and very low at P9.

Rich set: Air: Low at P6, medium at P7, low at P2, and medium at P8,
Temperature: Very high at P6, P7, P9, and low at P1,
Hardness: Low at P8, and very low at P9, P4, and P6,
Flow: High at P1, P5, P6, and P8.

Poor set: Air: Low at P5 and Medium at P9
Temperature: Very high at P5 and high at P8
Hardness: Low at P7 and very low at P4
Flow: High at P4 and P9.

The output produced by MVDS during the process undertaken with the above ideal set of measurement is reproduced on the next page. This is the only output included in this chapter, for conciseness. The outputs from all the other diagnostic processes undertaken with MVDS are given in appendix B.

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(very_high, temp, p6, []),
 value(very_high, temp, p7, [])].

------ Predictions obtained about entity [temp] -----

value(high,temp,p7,[r1])
value(very_high,temp,p5,[p])
value(high,temp,p8,[r2])
value(medium,temp,p8,[r1,r2])
value(medium,temp,p4,[p,b2])
value(very_high,temp,p9,[r1,r2,b2])
value(very_high,temp,p9,[p,b2,v])
value(medium,temp,p4,[r1,r2,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p7 : temp = very_high vs temp = high
 (resulting conflict set: [r1] )
Symptom at loc. p8 : temp = high vs temp = medium
 (resulting conflict set: [r1,r2] )

------ Updated minimal conflict sets --------
[r1]

---------- Updated minimal candidates ----------
[r1]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
Entity <flow> put under focus because local influence
between <flow> and <temp> (involved in conflicts),
within component r1

Entity <air> put under focus because local influence
between <air> and <temp> (involved in conflicts),
within component r1

*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |:  [value(high, flow, p5, []), value(high, flow, p6, [])].

Enter the measurements for entity air, with format:
[value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(low, air, p6, []), value(medium, air, p7, [])].

------- Predictions obtained about entity [flow,air] -----

value(low,air,p7,[r1])
value(low,air,p5,[p])
value(medium,air,p6,[r1])
value(medium,air,p8,[r2])
value(high,flow,p4,[b2])
value(high,flow,p6,[p])
value(high,flow,p7,[r1])
value(high,flow,p5,[p])
value(low,air,p8,[r1,r2])
value(low,air,p4,[p,b2])
value(medium,air,p5,[r1,p])
value(medium,air,p9,[r2,b2])
value(high,flow,p9,[b2,v])
value(high,flow,p7,[p,r1])
value(high,flow,p8,[r1,r2])
value(high,flow,p4,[p,b2])
value(low,air,p9,[r1,r2,b2])
value(low,air,p9,[p,b2,v])
value(medium,air,p4,[r1,p,b2])
value(medium,air,p4,[r2,b2,v])
value(high,flow,p8,[p,r1,r2])
```

value(high,flow,p9,[r1,r2,b2])
value(high,flow,p9,[p,b2,v])
value(low,air,p4,[r1,r2,b2,v])
value(medium,air,p9,[r1,p,b2,v])
value(high,flow,p9,[p,r1,r2,b2])
value(high,flow,p4,[r1,r2,b2,v])
value(high,flow,p4,[p,r1,r2,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p6 : air = low vs air = medium
(resulting conflict set: [r1])
Symptom at loc. p7 : air = medium vs air = low
(resulting conflict set: [r1])
Symptom at loc. p5 : air = low vs air = medium
(resulting conflict set: [r1,p])
Symptom at loc. p8 : air = medium vs air = low
(resulting conflict set: [r1,r2])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [p,r2,b2,v])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [r2,p,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r1,p,b2,v])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2,v])

------- Updated minimal conflict sets --------
[p,r2,b2,v]
[r1]

---------- Updated minimal candidates ----------
[p,r1]
[r2,r1]
[b2,r1]
[v,r1]

**** Result assessment ****
> Global influences:
Entity <hard> put under focus because global influence
between <hard> and <air> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format:
[value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(low, hard, p8, []), value(very_low, hard, p9,
[])].

------- Predictions obtained about entity [hard] -----

value(low,hard,p9,[b2])
value(low,hard,p7,[r2])
value(very_low,hard,p8,[b2])
value(very_low,hard,p4,[v])
value(low,hard,p4,[b2,v])
value(low,hard,p6,[r2,r1])
value(very_low,hard,p7,[b2,r2])
value(very_low,hard,p5,[v,b2])
value(low,hard,p5,[r2,r1,p])
value(very_low,hard,p6,[b2,r2,r1])
value(very_low,hard,p6,[v,b2,p])
value(low,hard,p4,[r2,r1,p,b2])
value(very_low,hard,p5,[b2,r2,r1,p])
value(very_low,hard,p7,[v,b2,p,r1])
value(low,hard,p9,[r2,r1,p,b2,v])
value(very_low,hard,p8,[v,b2,p,r1,r2])

---------- New symptoms and conflict sets ----------

```
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [b2] )
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p,r1,r2] )
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [b2] )
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2,v] )
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2] )
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r2,v,b2,p,r1] )
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [b2,v] )
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [v,r2,r1,p,b2] )
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1] )
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,v,b2,p] )
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1,p] )
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1,p] )

-------- Updated minimal conflict sets --------
[r1]
[b2]

---------- Updated minimal candidates ----------
[r1,b2]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.
The entities involved in new conflicts are not causally linked
        => End
*** End of result assessment ***
*** End of process ***
```

The above diagnostic process performed by MVDS follows an efficient path of reasoning, where the minimal candidates are successively refined from the single {r1} to the set of candidates {r1, r2}, {r1, b2}, {r1, p} and {r1, v}, and to the final minimal candidates {r1, b2}. As explained in section 5.5.4, this represents a satisfactory diagnosis. The diagnoses obtained by MVDS, or with sets of available entities restricted to {temp} or {hardness}, are shown in table 5.1.

- Scenario 1 - Satisfactory diagnosis: {r1, b2}	Final minimal candidate(s)		
Type of set of measurements	MVDS: all entities available	Entities available: {temp}	Entities available: {hardness}
Ideal	{r1, b2}	{r1}	{b2}
Rich	{r1, b2}	{r1,b2} {r1,r2}	{r1, b2} {r2,b2}
Poor	{r2,b2} {r1,b2} {r1,v} {p,b2} {p,v}	{r2} {r1} {p}	{r2,p} {r2,r1} {v,p} {v,r1} {b2}

Table 5.1 - Diagnosis results obtained with scenario 1

The analysis of these results is presented in chapter 6.

5.5.6 Diagnostic processes performed on scenario 2

In the same manner as for scenario 1, three sets of measurements are used for diagnosing scenario 2. These sets are used for comparing the performance of MVDS using a complete set of available entities {flow, pressure, temp, air, hardness} with the performance obtained with a set of available entities restricted to {temp}, or {flow}, or {hard}.

The sets of measurements used are listed below.

Ideal set:
Flow: High at P5, and medium at P6.
Temp: Very high at P6 and P7.
Hardness: Low at P6 and very low at P7.
Air content: Very low at P6 and P8.

Rich set:
Flow: High at P2 and P5, and medium at P6 and P8.
Temp: Medium at P4, high at P8, and Very high at P6 and P7.
Hardness: Low at P4 and P7, very low at P7 and P8.
Air content: Very low at P4, P5, P7 and P9.

Poor set:
Flow: High at P4, and medium at P7.
Temp: Very high at P6 and high at P8.
Hardness: Low at P4 and very low at P9.
Air content: Very low at P4 and P8.

The results obtained are shown in table 5.2.

- Scenario 2 - Satisfactory diagnosis {r1,p}	Final minimal candidate(s)			
Type of set of measurements	MVDS: all entities available	Entities available: {temp}	Entities available: {flow}	Entities available: {hardness}
Ideal	{r1, p}	{r1}	{r1,p}, {r2,p}, {b2,p}, {v,p}	{p,r1}, {r2,r1} {b2,r1}, {v,r1}
Rich	{p,r1,v}, {p,r1,b2}	{v,r1}, {b2,r1}	{b2,p}, {v,p}	{r1,v}, {r1,b2}
Poor	{r1,v} {r2,p,v} {r2,b2,v}	{r2}, {r1}	{r2,p} {r2,r1} {v,p} {v,r1} {b2}	{b2,v} {r2,v} {r1,v} {p,v}

Table 5.2 - Diagnosis results obtained with scenario 2

The analysis of these results is presented in chapter 6.

5.6 Conclusion

A small example and a larger case-study have been described and used as test-beds for MVDS.
In order to be able to evaluate the advantage of including all the relevant entities to a system into the set of available entities, MVDS is used either with this exhaustive set, or with sets restricted to a single entity.

Because the set of measurements used in a diagnostic process influences the result, the fairness of comparisons can only be obtained if several sets of measurements are used in a series of tests. Three types of sets of measurements have been identified, namely the types Ideal, Rich, or Poor.

Following these test conditions, one fault scenario has been diagnosed for the air-conditioning example, and two scenarios for the hot-water and heating system. The air-conditioning example has demonstrated that MVDS reached a satisfactory diagnosis, whereas the other systems have not. For each of the two scenarios on the hot-water and heating system, the diagnostic results have been reported and are the subject of an analysis presented in chapter 6.

Chapter 6

Analysis of MVDS' diagnostic performance

6.1 Introduction

The analysis of the performance of MVDS is undertaken in a comparative manner. Section 6.2 describes the theoretical framework of these comparisons, which are presented in the following sections.

The diagnostic performance of MVDS, when it is applied to the air-conditioning system described in chapter 5, is simple. It exhibits a clear advantage over the diagnostic performances undertaken using an incomplete set of available entities, since it reaches the satisfactory diagnosis whereas the other performances do not. In section 6.3, although mentioned above as simple, the air-conditioning example is considered and explained, in order to see how it demonstrates the advantages of MVDS.

On the more complex case-study, however, the various performances obtained on the two scenarios with the different sets of available entities need to be analysed more attentively. This analysis is the object of section 6.4, where the diagnostic performances on the two fault scenarios set up in chapter 5, on the hot-water and heating system, are presented.

The penultimate section evaluates the quality of the traces produced by MVDS during the diagnosis processes.

6.2 Basis for the comparative evaluation of MVDS

The performance of MVDS is evaluated in a comparative manner. However, no existing fault-diagnosis system provides any way of adapting the set of investigated entities to the task at hand, thus, MVDS cannot be compared directly to a particular system.

For every consistency-based fault-diagnosis system, the set of investigated entities is fixed at the start of the task. It is an initial condition of the task. Thus, the same system can be run using a system's representation including only one entity, or including a few entities, or including all the entities that are relevant to the physical system's functioning.

Depending on the set of entities included in the system's representation at the start of the task, the process performed with an existing fault-diagnosis system varies in accuracy and in cost. The performance of MVDS does not depend on this initial set of entities, since it is adapted automatically to an efficient set during the process. Therefore, the performance of MVDS cannot be compared directly to another system's, but can be compared with the performance that would be obtained using different fixed sets of investigated entities. In that way, it is possible to evaluate the advantage of an automatic adaptation of the set of investigated entities to the task at hand.

In order to make these comparisons, MVDS is run several times on the same fault-scenario, using the same sets of measurements, but with different sets of entities investigated. First, it is run with an adaptable set. This is the process that is really obtained by the strategy in MVDS. Secondly, this adaptability is disabled in MVDS, and the set of entities investigated is fixed to either a set with only one entity, or a set including all the entities that are relevant to the functioning of the physical system. This second set of performances represents generically the performances of existing systems with different initial conditions. As a consequence, it is possible to evaluate the differences between the diagnosis processes obtained with an adaptable set of investigated entities and obtained with various fixed sets of investigated entities.

Comparing the different diagnosis processes is undertaken by comparing two factors: the accuracy of the result, and the cost of the process.

6.2.1 Accuracy

The accuracy of the result is evaluated using the concept of satisfactory diagnosis defined in section 5.3.5.1. Each fault-scenario is associated with a satisfactory diagnosis.

The set of all the possible diagnoses is not fully ordered by accuracy, but common-sense provides this set with a partial order. This simple ordering follows these principles (or rules):

- Rule 1: if the set of minimal candidates is reduced a single minimal candidate C, and if C is the scenario's satisfactory diagnosis, then the accuracy of this result is maximum.
- Rule 2: if a set S1 of minimal candidates contains the scenario's satisfactory diagnosis S, then S1 is more accurate than any set of minimal candidates which does not include S.
- Rule 3: if two sets of minimal candidates both contain the scenario's satisfactory diagnosis, then the one with the smaller number of minimal candidates is more accurate.
- Rule 4: if a set S2 of minimal candidates contains a superset of the scenario's satisfactory diagnosis S, then S2 is more accurate than any set of minimal candidates which does not include a superset of S.
- Rule 5: if two sets of minimal candidates both contain a superset of the scenario's satisfactory diagnosis, then the one with the smaller number of minimal candidates is more accurate.

In the analysis of the case-study, the accuracy of the result obtained with MVDS is only compared to the accuracy of the result obtained with a too limited fixed set of investigated entities. This is because the accuracy of the result obtained with a complete fixed set of entities is equal to the accuracy obtained with MVDS, since the extra investigations of the former compared to the latter are investigations of correct entities, therefore not holding any information for a consistency-based approach.

6.2.2 Cost

The cost of the process using MVDS needs only to be compared with the cost of the processes which have achieved the same accuracy.

The cost of the different diagnosis processes include several cost elements. With a fixed set of investigated entities or with the strategy of MVDS, the total cost always consists of the computational cost and the measurement cost. Evaluating the computational cost must take into account the number of entities investigated, and, for MVDS, the extra-computational cost incurred by the result assessment (search through causal knowledge if new symptoms are present) and the dynamic revisions (modification of the system's representation). The measurement cost is evaluated by the number of investigated entities. This is because each investigation of an entity requires the acquisition of a set of measurements.

It is important to notice that, in non-monitored systems, making measurements is an expensive process. On the contrary, the high computational speed of modern computers reduces the importance of the computational cost. As a consequence, for many applications, the reduction of the measurement cost is welcome even if it is accompanied with a increase of the computational cost. Thus, the differences between incurred measurement costs is argued to be more important than the differences between incurred computational costs.

6.3 A clear case: the air-conditioning example

The structure of the 3-piece air-conditioning example is recalled on the graph in figure 6.1 below. For a detailed description of the system, its modelling, and the fault scenario considered, please see section 5.3.

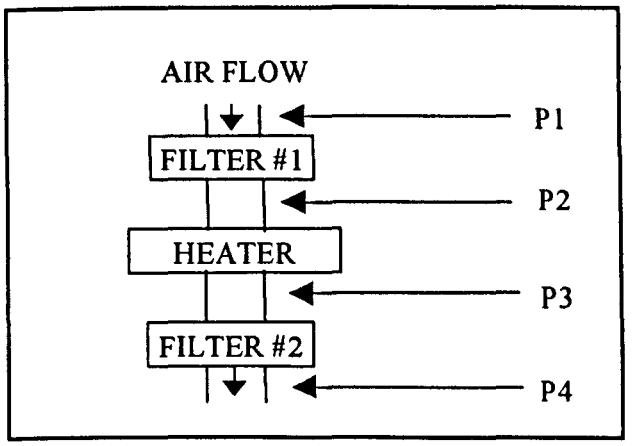


Figure 6.1 – Recall of the 3-piece air-conditioning system

The complex multiple-fault situation occurring in the fault-scenario contains two faulty components, the heater and filter #1, whose faults affect two entities, 'dust' and 'temperature'.

6.3.1 Accuracy of the results

The fault-scenario's satisfactory diagnosis is {Heater, Filter #1}. The final minimal candidates obtained with the three different sets of available entities are recalled below.

	Sets of investigated entities			
	Fixed to {Dust}	Fixed to {Temperature}	Adapted by MVDS to {Flow, Temperature, Dust}	Fixed to {Flow, Temperature, Dust, Water content}
Final minimal candidate	{Filter #1}	{Heater}	{Heater, Filter #1}	{Heater, Filter #1}

Table 6.1 - Diagnosis results obtained on the air-conditioning example

Only a single faulty component is diagnosed when the set of available entities is fixed to {Dust} or {Temperature}, but the two faulty components are diagnosed if MVDS is used with a adaptable set of available entities.

The comparison of these results is clear. By rule 1, the accuracy result obtained by using the adaptable set of available entities, *i.e.* using the strategy in MVDS, is maximum. By rule 2, this result is more accurate than the two results obtained with a single investigated entity, since neither of these latter results include the satisfactory diagnosis. On this air-conditioning example, MVDS has therefore obtained a more accurate diagnosis than approaches using a too restricted set of investigated entities.

Similarly to the result obtained with MVDS, the accuracy of the result obtained with a fixed complete set of investigated entities is maximum.

6.3.2 Cost of the accurate processes

Computational cost:
In the case of a system where the complete set of available entities is put under focus at once, the number of entities investigated is the number of elements in the set of available entities. In the present case of the air-conditioning example, this number is four. In the case of MVDS, the number of entities investigated is three. However, two dynamic revisions and two result assessments (with search of the causal knowledge) have been conducted, as it can be seen on the output in section 5.3.5.2. MVDS has therefore avoided some unnecessary computational cost by avoiding the investigation of one entity, but had to bear extra computational cost for conducting the four actions mentioned before.

Measurement cost:
Because the process using MVDS has investigated three entities and the process using a fixed complete set of entities has investigated four entities, the measurement cost has been cut down by the use of the strategy in MVDS.

6.3.3 Quality of the output

Beside the quantitative advantages consisting of the diagnosis precision and its low cost, it is important to notice the clarity of the output in section 5.3.5.1. It has been possible, by using the GDE method for conflict and candidate generation, to produce an explicit trace close to common-sense reasoning. English sentences are widely used in the rich trace that can favour human-machine co-operation and has an important explanatory value, as explained in section 3.4.1.3.

6.3.4 Conclusion on the air-conditioning example

This simple example shows that it is possible to investigate the entities independently of each other, and still to combine the conflict sets to reach a precise diagnosis taking into account the various entities. It has also shown that the measurement cost of obtaining the satisfactory diagnosis can be significantly reduced in comparison to systems where all the entities available are investigated at once.

Furthermore, because the process is split into several sub-parts, and because common-sense algorithms are used, the output produced by MVDS on this example has been shown to be favourable to co-operative work and to be significantly self-explanatory.

However, the evaluation of the performance of MVDS on the more complex case-study using the hot-water and heating system is more complicated than for this small example. This evaluation is the subject of the two next sections.

6.4 Analysis of the diagnostic performances on the case-study

The two scenarios 1 and 2 set up for the case-study have been described in section 5.5.4. Recall briefly their main characteristics. In scenario 1, the temperature is the base-entity and the components that need repairing are the radiator r1, in order to stop the air intake, and the boiler b2 that needs to be de-scaled in order to regain its normal efficiency. {r1, b2} is therefore the satisfactory minimal candidate. In scenario 2, the temperature is the base-entity again, and the faulty components are the pump p, since it needs to be replaced (or the seals changed), and the radiator r1, since it needs to be replaced or emptied of the sludge. {r1, p} is therefore the satisfactory minimal candidate.

The diagnostic performances undertaken on these two scenarios are analysed by looking at the accuracy of the diagnoses obtained and by looking at the cost incurred.

6.4.1 Accuracy of the results

For every diagnostic engine, the accuracy of the result depends on the set of entities investigated and also on the set of measurements used. As explained in chapter 5, three different sets of measurements have been used on each scenario. The two tables with the different diagnoses obtained, with respect to the set of available entities and the set of measurements, are reproduced below for convenience.

- Scenario 1 - Satisfactory minimal candidate: {r1, b2}	Final minimal candidate(s)		
Type of set of measurements	MVDS: all entities available	Entities available: {temp}	Entities available: {hardness}
Ideal	{r1, b2}	{r1}	{b2}
Rich	{r1, b2}	{r1,b2}, {r1,r2}	{r1, b2}, {r2,b2}
Poor	{r2,b2}, {r1,b2}, {r1,v}, {p,b2}, {p,v}	{r2}, {r1}, {p}	{r2,p}, {r2,r1}, {v,p}, {v,r1}, {b2}

Table 6.2 – Recall of diagnosis results obtained with scenario1

- Scenario 2 - Satisfactory minimal candidate {r1,p}	Final minimal candidate(s)			
Type of set of measurements	MVDS: all entities available	Entities available: {temp}	Entities available: {flow}	Entities available: {hardness}
Ideal	{r1, p}	{r1}	{r1,p}, {r2,p}, {b2,p}, {v,p}	{p,r1}, {r2,r1} {b2,r1}, {v,r1}
Rich	{p,r1,v}, {p,r1,b2}	{v,r1}, {b2,r1}	{b2,p}, {v,p}	{r1,v}, {r1,b2}
Poor	{r1,v}, {r2,p,v}, {r2,b2,v}	{r2}, {r1}	{r2,p}, {r2,r1}, {v,p}, {v,r1}, {b2}	{b2,v}, {r2,v}, {r1,v}, {p,v}

Table 6.3 – Recall of diagnosis results obtained with scenario 2

6.4.1.1 Using the ideal set of measurements

Investigating a single entity in scenario 1 results in the diagnosis of one faulty component out of the two present. MVDS reaches the satisfactory diagnosis {r1, b2} as the only minimal candidate. By rule 1, the accuracy is maximum, and by rule 2, it is higher than the accuracy achieved by the processes which use a single entity. This case is similar to the air-conditioning example, where the set of measurements used is an ideal set. It confirms the efficiency of MVDS when an ideal set of measurements is provided.

In scenario 2, investigating only the entity 'temperature' results again in the diagnosis of only one fault. Investigating only the entity 'flow' or only the entity 'hardness' provides four minimal candidates, showing a poor discrimination of candidates. However, one of the minimal candidates obtained with the entity 'hardness' is the satisfactory minimal candidate. However, it must be mentioned that this finding is a coincidence, since no relevant symptom for the air-intake in the pump can be found by investigating hardness measurements, which, by nature, do not hold any information about air contents. Furthermore, recall that this minimal candidate is not discriminated from three other ones. The diagnosis achieved by MVDS is still the satisfactory diagnosis, thus its accuracy is maximum. By rule 2, it is more accurate than the results obtained by the processes using the fixed sets {temperature} and {Flow}. By rule 3, MVDS achieves a higher accuracy than the process using the fixed set {Hardness}.

6.4.1.2 Using the rich set of measurements

In scenario 1, each of the two processes investigating a single entity reaches two final minimal candidates, including the satisfactory minimal candidate. It is again by coincidence that this satisfactory minimal candidate is obtained, since each of the single entity processes misses out on some necessary reasoning to diagnose both of the occurring faults. This is confirmed by the performances of the single-entity processes in scenario 2, where the satisfactory minimal candidate is not obtained by any one of the three.

In scenario 1, MVDS reaches the satisfactory minimal candidate as the only minimal candidate. Therefore, the accuracy is maximum (by rule 1), and higher than the accuracy achieved by using a single investigated entity (by rule 3). In scenario 2, where the single-entity processes obtain two

minimal candidates of two components each, MVDS obtains two minimal candidates of three components. The discrimination is therefore higher. This is to be expected, since having more measurements can only increase the discrimination. However, notice the efficiency of the extra-discrimination, since each of the two minimal candidates from MVDS contains the satisfactory minimal candidate: {p,r1,v} and {p,r1,b2} both contain {p, r1}. Even if these minimal candidates are not the satisfactory minimal candidate, they are useful candidates since they might indicate to the operator that the components 'pump p' and 'radiator r1' are highly suspicious. By using the rules for accuracy ordering, the accuracy achieved by MVDS is higher than the accuracy of the other results by rule 4.

6.4.1.3 Using the poor set of measurements

Using scenario 1, MVDS performs efficiently. The single-entity processes reach either three single-component minimal candidates including only one of the faulty components, or five minimal candidates of one or two components, where the satisfactory minimal candidate is not included. MVDS, however, obtains five minimal candidates of two or three components, out of which one is the satisfactory minimal candidate. The discrimination has therefore been efficient. The accuracy of the result obtained using MVDS is higher than the accuracy of the other results, by rule 2.

Using scenario 2, the results of the single-entity processes are still unsatisfactory. Investigating the entity 'temperature' reaches two single-component minimal candidates, out of which only one is a faulty component. When 'flow' or 'hardness' are the only investigated entity, respectively five and six minimal candidates of one or two components are obtained, which do not include the satisfactory minimal candidate. The result obtained by MVDS consists of three minimal candidates, one of two and two of three components. As opposed to scenario 1, the discrimination has not been efficient, since none of these three minimal candidates is equal to, or a superset of, the satisfactory diagnosis. No real progress has therefore been achieved by investigating further entities. This is because the poor quality of the set of measurements does not enable useful conflict sets to be discovered. The different results have the same accuracy, since no rule can be used to order them.

6.4.1.4 Conclusion on the accuracy

The results obtained across the different sets of measurements for the two scenarios differ widely. Investigating all the entities affected by a fault has sometimes allowed MVDS to isolate the satisfactory minimal candidate as the single final minimal candidate, but sometimes resulted in not much progress. The former case is true with the ideal sets of measurements, and the latter with the poor set of measurements in scenario 2. With the poor set of measurements in scenario 1, the increase of accuracy is present in comparison with the single-entity processes, but is not as dramatic since the satisfactory minimal candidate is listed but not singled out.

The ability of MVDS to reach a highly accurate diagnosis is therefore proportional to the quality of the set of measurements. This quality is always a necessary condition for an efficient diagnosis, but these tests prove that it is also a sufficient condition for MVDS. Indeed, with an efficiency that is proportional to the quality of the sets of measurements, MVDS manages to combine together investigations of different entities in order to refine the minimal candidates towards the satisfactory minimal candidate. MVDS always performs as well as, or better than, other systems.

6.4.2 Stability

The stability of a diagnosis strategy is defined in this thesis as the stability of its results' quality across different fault situations and available sets of measurements.

Out of the six diagnostic tasks against which MVDS has been tested:

- three results are the satisfactory minimal candidate singled out,
- two results consist of minimal candidates including the satisfactory minimal candidate or supersets of it.
- one result does not include the satisfactory minimal candidate nor any supersets of it.

The important point is to look in MVDS for any loss of stability from systems using a fixed complete set of entities under focus. In view of the results above, the stability of MVDS is satisfactory, since it

still obtains useful results when the sets of measurements are changed for sets of lower quality, or when using different scenarios.

6.4.3 Cost of the accurate processes

The cost of obtaining the diagnosis with MVDS must be compared with the cost of obtaining the diagnosis by investigating all the entities available at once.

In the hot-water and heating system of the case-study, there are five entities identified as available: pressure, flow, temperature, air content, and hardness. In all the diagnostic tasks performed by MVDS, only four entities were investigated.

Computational cost:
Using MVDS on the hot-water and heating system, the computational cost of investigating one unnecessary entity was avoided. It added the computational cost of two dynamic revisions and three result assessments, as it can be seen on the outputs in chapter 5 and in appendix B.

Measurement cost:
Since it avoided the unnecessary investigation of one entity, the process using the approach in MVDS is associated with a lower measurement cost than the process using a complete fixed set of investigated entities.

Notice that the set of available entities is the set of the entities which are involved in the model of the components. Thus, adopting more complex models would involve more entities and increase the size of the set of available entities. However, the number of entities involved in the fault-scenarios treated in this work remains at four. The greater the number of available entities not affected by one complex multiple-fault situation, the greater the amount of measurement work that MVDS can potentially save. If the same fault-scenarios were studied with more complex models including more available entities, the decrease of the measurement cost could therefore be stronger.

6.5 Output produced by MVDS

In chapter 3, the importance of producing a clear trace of the process has been stressed, for its role in human-checking, for its necessity in co-operative work, and for its explanatory value. The traces of all the diagnostic processes performed by MVDS are printed in Appendix B. They show that the trace is clear and rich, since it features:

- listings of actions undertaken, in English,
- explanations of dynamic revisions undertaken, in English,
- listings of intermediate results as predictions, symptoms and conflict sets.

6.6 Conclusion

The performance of MVDS at diagnosing one fault scenario on the air-conditioning system and two fault scenarios on the hot-water and heating system have been analysed.

In comparison with systems where a single entity is investigated, the discrimination obtained in MVDS by combining together independent investigations of several entities is noticed to be sound and efficient. Indeed, the minimal candidates obtained by MVDS either consist of the satisfactory diagnosis for the fault-scenario, or are closer to this satisfactory diagnosis than the candidates obtained by investigation of a single entity. This efficiency of the discrimination is noticed to be stable across different scenarios and sets of measurements, even if it decreases when the quality of the sets of measurements decreases.

In comparison with a system where all the entities available are investigated together at once, and therefore reaching the same accuracy, the measurement cost is reduced by the use of MVDS. This is because the models have been reduced to the smallest set of entities necessary to describe the function of each component.

Comparing the computational costs is not generally possible. Extra-computational cost is created in MVDS by the occurrence of result assessments and dynamic revisions, but some computational cost is removed by avoiding the investigation of unnecessary entities. However, it is important to note that the

computational cost generated by any of the approaches is low. The computational time involved in running any of the diagnostic processes on the case-study has an order of magnitude of a few seconds. On the contrary, the measurement cost on non-monitored hot-water and heating systems is high, since making measurements on non-monitored physical system is a long and expensive process. Therefore, the most important conclusion of the analysis presented in this chapter is that the use of the approach in MVDS results in a decrease of the measurement cost.

The output trace printed by MVDS during the diagnostic process is clear, rich, and self-explanatory, thereby proving the usefulness of choosing the GDE method for conflict and candidate generation because of its common-sense reasoning characteristic. The trace could even be made clearer, to a user who would be unaware of the underlying diagnostic process, by suppressing the references to terms such as symptoms or candidates. However, the current format of the trace is satisfactorily clear and self-explanatory.

Chapter 7

Conclusions and further work

7.1 Introduction

This chapter reviews the whole research and suggests areas of further work. The first section summarises the thesis by outlining its major points. The limitations and contributions of this research are listed respectively in sections 7.3 and 7.4. These limitations often result in suggestions for further work, which are described in the final section of this thesis.

7.2 Review

The complexity of physical systems results in the potential for complicated diagnosis tasks to be undertaken. These tasks include the diagnosis of complex multiple-fault situations, characterised by several of the system's behavioural entities being affected by faults. These situations are likely on complex physical systems because they involve many entities in their behaviour and because these entities influence each other's behaviours. A physical system is specifically at risk if it is poorly or not monitored, if it needs regular maintenance, if it is subject to variable and uncontrolled conditions of use, or if it contains components with partially known behaviours and ageing processes.

The investigation, i.e. simulation for consistency-checks and conflict collection, of more than one entity is necessary when addressing a complex multiple-fault situation, since several entities are affected by faults. However, investigating all the available entities can include some unnecessary work, since the entities that are not affected by any fault do not need to be investigated. The problem is that it is not possible to know beforehand which are the entities affected by a fault. Therefore, with existing diagnosis systems, the set of investigated entities would typically either be reduced to the entity that externally manifested a fault, or contain all the available entities. Depending on this set of investigated entities, fault-diagnosis systems either reach an incomplete or inaccurate result, or achieve accuracy at a high cost.

This work proposes a novel diagnosis strategy, called MVDS for Multiple-Variable Diagnosis Strategy, that is specifically designed to tackle the issue of diagnosing complex multiple-fault situations. Since, as mentioned above, the set of entities that need to be investigated cannot be known before the diagnostic process, it is to be found progressively along this process.

In order to do so in MVDS, the different entities involved in a physical system's behaviour, namely the available entities, are modelled independently. It is then possible to investigate any one of them independently. An investigated entity is said to be 'under focus'. The information obtained by independent investigations of the entities under focus is combined together to form the overall result.

Causal information is used to decide which entities need to be under focus. It consists of a declaration of the known influences between entities and quantities. This causal information is considered together with the latest intermediate results obtained in the current diagnostic process, so that the set of entities under focus is dynamically updated all along the process.

Using MVDS to adapt the set of entities under focus to the task at hand, by using the most recent intermediate results available, obviates the need for the unnecessary investigation of unaffected entities. By reducing the number of entities that are being investigated, MVDS reduces the computational costs that would have been necessary to check whether these entities are faulty. This reduction in computational cost is not achieved at the expense of accuracy. On the contrary, the approach used by MVDS ensures that all the entities that matter are investigated.

The use of qualitative reasoning for the whole process and the use of GDE's method for candidate generation have resulted in a clear and rich trace that provides the human operator with a good insight into the process. The sectioning of the process into investigations of different entities is argued to be closer to human reasoning than sectioning using the changes of the algebra's accuracy or of the model's granularity. Thus the potential for co-operative work is increased.

The performance of MVDS has been tested on a small air-conditioning example with one fault scenario, and on a larger case-study of a hot-water and heating system with two fault scenarios. The test on the air-conditioning example has illustrated clearly how MVDS can reach the satisfactory minimal candidate without unnecessary work. On each scenario of the case-study, three different sets of measurements are used: an ideal set, a rich set and a poor set, in decreasing order of quality. In comparison with the results obtained with only one entity under focus, MVDS' results are demonstrated to be more accurate. However, the gain in accuracy, *i.e.* the efficiency of the discrimination resulting from further investigations, decreases with the quality of the sets of measurements. Despite this, a satisfactory stability of the results' accuracy is noticed. The measurement cost incurred in MVDS' processes is reduced, in comparison with systems investigating all the available entities at once. This ratio is dependent on the models used for the components, but these figures are argued to be minimal. Finally, tests on MVDS have demonstrated the clarity and completeness of the trace produced during each process.

7.3 Limitations

The current state of MVDS has limitations of different types. Some are concerned with the assumptions made, some are concerned with the algorithm's efficiency, and some are concerned with the use of MVDS in an industrial application. Other types of limitations are related to the analysis of MVDS.

7.3.1 Modelling assumptions

The limitation caused by the modelling assumptions made to build MVDS is the availability of the models needed, since the strategy uses independent models of the different entities involved in a

physical system's behaviour. Obtaining these models is possible through the use of functional modelling, as explained in section 3.4.2.1, but could prove to be difficult in certain cases, where, for example, the function of a certain component involves two entities that cannot be detached from each other. This limitation is linked with the limitations of functional modelling.

7.3.2 Diagnosis system's incompleteness

MVDS is limited in its performance by the fact that it is only a strategy, and not a complete fault-diagnosis system. In its present form, the diagnosis system using MVDS is weakened by its lack of tools. For example, MVDS does not include a module for dealing with the need for changing the algebra or changing the granularity of the model, nor does it include a measurement selection tool. MVDS is a strategy which has been built to be integrated into a fault-diagnosis system together with other performing modules. Therefore, it should not be considered as a complete fault-diagnosis system.

7.3.3 Knowledge availability and quality

Another limitation related to the assumptions made to build MVDS is concerned with the causal knowledge used. The strategy relies on the use of causal information, which can be limited in different ways. The knowledge of the influences between quantities and entities needs to be available, as completely as possible, and as correctly as possible. However, in comparison with rule-based systems, the quality of this shallow knowledge is not as critical. Indeed, causal knowledge is only used in MVDS as a guide for trying to reach a good balance between accuracy and cost of the diagnosis. If the causal knowledge was to be incomplete or corrupted, the process would lose efficiency, but because the causal knowledge is not used for simulations and candidate generation, the soundness of the diagnosis would remain unaffected. The completeness and correctness of the causal knowledge is therefore a limitation of MVDS' efficiency but not correctness.

7.3.4 Co-operation

The co-operative ability in MVDS is reduced to a token gesture: asking whether or not the operator wants the process to continue, acknowledging thereby the possibility that the operator possesses some knowledge which would render further investigations unnecessary. Implementing this interaction has proved that a diagnostic process following MVDS' strategy is split into sub-parts whose nature, being centred on entities, makes it favourable for co-operative work. However, the co-operative ability of MVDS, in its current development, is limited by its restriction to a single type of user intervention, as recalled above.

7.3.5 Different sets of measurements

The study of the efficiency of MVDS is limited in its depth by the importance of the influence of the set of measurements on the corresponding diagnostic process. Indeed the variety of sets of measurements is so wide that three sets of measurements on each scenario of the case-study are not sufficient to represent the whole range of potential diagnostic situations. This limitation can be reduced but not removed. It can be reduced by increasing the number of sets of measurements against which MVDS is tested, but it cannot be removed since the entire set of possible sets of measurements is too large to be covered.

These limitations can be helped by some specific work, therefore they are all associated with a suggestion of further work. These suggestions are described in section 7.5.

7.4 Contributions

With the aim of investigating the management of a physical system's different entities for optimising their use in the fault-diagnosis of complex multiple-fault situations, this research has made the following contributions:

- Complex multiple-fault situations are formally defined, and their diagnosis, identified as a problematic fault-diagnosis task, is specifically addressed for the first time. A diagnosis strategy, called MVDS, has been designed for diagnosing complex multiple-fault situations.

The possibility of occurrence of complex multiple-fault situations on complex physical systems has been shown. It has been explained that the impossibility of knowing beforehand what are the entities affected by the faults requires a specific strategy where the set of entities investigated can be adapted.

- In order to be able to evaluate how prone to complex multiple-fault situations a physical system is, four risk characteristics have been identified.

These characteristics are: not being monitored, needing regular maintenance actions, being subject to variable and uncontrolled conditions of use, and containing components of partially unknown behaviour and ageing process. The more a physical system matches these characteristics, the more likely it is to develop complex multiple-fault situations. In this case, MVDS is an appropriate choice of strategy for their diagnosis.

- MVDS uses a library containing models of the different entities in the physical system. This organisation around the entities is a novel structure for the library of models.

This structure, based on entities, is needed in order to optimise the diagnosis of complex multiple-fault situations. At a higher level, there is still a more usual component-based structure, but each component is modelled through the behaviour of each relevant entity within it.

- The strategy uses the objects 'entity' and 'quantity'. They are precisely defined, as opposed to their usual grouping under the name 'variable'. The causal knowledge about the physical system is structured with respect to these objects.

An entity refers to a physical entity, which is free of any concept of location. When a location in the system is present in the object, then it is named as a quantity. The reason for this distinction is that an entity cannot be evaluated, since it refers to a flow of information, as opposed to a quantity which has a value. The causal knowledge includes influences between entities, therefore valid everywhere in the physical system, and influences between quantities, therefore valid at certain locations only. Influences

between entities are called global influences, and influences between quantities are called local influences.

- The number of changes in the system's representation along the diagnostic process is potentially large and its limit is application-dependent. The formal definition of a 'diagnosis session' is proposed, referring to the reasoning process of obtaining a diagnosis candidate without any changes either to the physical system's representation or to the set of measurements.

The changes to the system's representation are concerned with updating the set of entities that need investigating, rather than with its structural accuracy or with the algebra's granularity. Thus, they are only limited in number by how many entities are relevant to the physical system's behaviour. In traditional systems, changes to the system's representation are typically limited to one or two occurrences. A diagnostic process following MVDS' strategy can therefore contain a large number of diagnosis sessions separated by representation changes.

- Three classes of measurement sets have been identified: ideal, rich, and poor. No classifications of possible measurement sets has been proposed before. The testing of MVDS is made against a range of measurement sets from these various classes, which is a novel way of evaluating a system's performance.

The performance of a diagnosis system depends strongly on the measurements available. Evaluating a diagnosis system should therefore take into account the variety of measurement sets that the diagnosis system might have to use. Even if a system incorporates a measurement selection module, it might still meet different qualities of measurement sets since the quality of the set of possible measurements varies with every physical system.

- Allowing co-operation and identifying its potential advantages are issues which are novel in the field of model-based fault-diagnosis.

In order to ensure that a process is usable on physical systems without appropriate diagnostic records, it must be based on deep-knowledge. However, because shallow knowledge is an important source of useful information, to increase efficiency, its use is a significant advantage. The integration of this knowledge, when it becomes useful during the task at hand, is possible through co-operation with the

operator. The common-sense characteristics of one strategy favour the use of co-operation. Since it is centred on entities, MVDS is such a strategy. This is because reasoning on entities is closer to common-sense reasoning than, for example, reasoning about the granularity of an algebra.

7.5 Further work

The limitations of MVDS and its study, described in section 7.3, can be lessened by further related work. Many of the suggestions below are therefore concerned with development work around MVDS. However, three development suggestions are extended to ideas for further research in the fault-diagnosis field in general, namely the generalisation of adaptability of diagnosis systems, the development of co-operative fault-diagnosis, and the investigation of strategy-dependent measurement selection. These three suggestions for further work are covered first.

7.5.1 Integration in a complete system, generalisation of adaptability

MVDS provides a strategy for an economical management of the physical system's entities, but is not a complete fault-diagnosis system. Development work could therefore be aimed either at building modules around MVDS, or at integrating MVDS into an existing fault-diagnosis system. A first important module that is needed is a measurement selection module, since it has been shown that the efficiency of MVDS depends strongly on the quality of the measurement sets, despite a satisfactory stability of the strategy. A second important development is to complete the range of possible changes undertaken in a dynamic revision. In the same dynamic way as the set of entities under focus is updated, the granularity of the models used for these entities, or the accuracy of the algebra, could all be potentially adapted by a dynamic revision. In comparison with the traditional systems, adapting the set of entities under focus is a progression that needs to be followed by more work towards high adaptability of fault-diagnosis tools to the current task.

7.5.2 Development of co-operative fault-diagnosis

Even though MVDS is enabled for co-operation, the set of possible co-operative actions needs to be increased. The incorporation of shallow knowledge held by the user is a factor that can influence the efficiency of the process drastically. How valuable is, for example, the knowledge possessed by a user that one component is very old while another is brand new? Furthermore, because of its benefits, co-operative systems are more likely to be used for important decision-making [Woods, 1985], and therefore to be applied in a wide range of industries. Co-operative fault-diagnosis is a promising research domain.

7.5.3 Further testing, strategy-dependent measurement selection

The identification of three characteristics of measurement sets is a first step towards a more systematic classification of measurement sets. MVDS needs to be tested against more numerous sets, in order to extract more information about what type of sets fits MVDS best. In general, measurement selection modules do not take into account the type of diagnosis strategy used [de Kleer & Brown, 1987]. This is a limitation of such modules. For example, a measurement selection module for MVDS should try and minimise the number of dynamic revisions in a diagnostic process. For the same four entities being put under focus, it would be better to have two dynamic revisions resulting in two entities being put under focus each time, rather than four dynamic revisions putting only one entity under focus each time. This is an example of how the influence of the measurement sets on a diagnostic process goes further than their influence on the result's discrimination. This complete influence would need investigating.

7.5.4 Modelling work

Using MVDS requires independent models of entities which, in this work, are assumed to be available. Obtaining these models for a wide range of application domains, such as electrical, mechanical, electronics, or hydraulic, can be time-consuming. The construction of a large library of models available for use by MVDS would be an important step towards the wide use of MVDS on different physical systems. Looking at obtaining these models in an automatic manner is also a possibility.

7.5.5 Causal knowledge derivation

Because MVDS uses causal knowledge about the physical system at hand, it must be ensured that this knowledge is available, with sufficient completeness and soundness. At least three approaches could be taken to this problem. The first approach would be to conduct some knowledge elicitation work. In this case, completeness and soundness are limited, since this approach is not applicable with a physical system that has not been sufficiently observed beforehand. The second approach would be to extract the causal knowledge from the algebraic model of the physical system, if it is available. This approach, taken in [Travé-Massuyès & Milne, 1997], requires a tool for automated knowledge extraction to be built, and cannot reach completeness since all the influences are not embedded in the algebraic equations. The third approach would be to combine the two first approaches, *i.e.* to derive a causal graph automatically and complete it with the result of some knowledge elicitation.

References

Blundell, A. J. (1982). *Bond Graphs for Modelling Engineering Systems*. Halsted Press: Chichester.

Bredeweg, B.; de Koning, K; Schut, C. (1994). Modelling the Influence of Non-Changing Quantities. In Working Papers of *QR'94, The Eighth International Workshop on Qualitative Reasoning about Physical Systems*, pp. 13-23. Nara, Japan.

Bradley, E. (1994). Automatic construction of accurate models of physical systems. In *QR'94, The Eighth International Workshop on Qualitative Reasoning about Physical Systems*, pp. 13-23. Nara, Japan.

Brown, J.S.; Burton, R.; de Kleer, J. (1982). Pedagogical, natural language, and knowledge engineering issues in SOPHIE I, II, and III. In Sleeman, D. and Brown, J.S., (eds.), *Intelligent Tutoring Systems*, 227-282. Academic Press: NY.

Chadderton, D. V. (1991). *Building services engineering*. Chapman & Hall, London.

Clancy, D. J. & Kuipers, B. (1994). Model decomposition and simulation. In *QR 94, the eighth international workshop on QR about physical systems*, 45-54, Nara, Japan.

Clarke, A. A. & Smyth, M. G. G. (1993). A Cooperative Computer based on the Principles of Human Cooperation. *International Journal of Man-Machine Studies 38*.

Coghill, G. M.; Chantler, M. J.; Shen, Q.; Leitch, R. R. (1997). Towards model switching for diagnosis of dynamic systems. In Proceedings of *SAFEPROCESS'97*, Hull, United-Kingdom.

Coiera, E. (1992). The qualitative representation of physical systems. *The Knowledge Engineering Review* 7(1), 55-77.

Console, L.; Torasso, P. (1990). Integrating Models of the Correct Behavior into Abductive Diagnosis. In Proceedings of *ECAI 90, The European Conference on Artificial Intelligence.*

Curd, E. F. & Howard, C. A. (1996). *Introduction to Building Services*. MacMillan Press Ltd.

Dague, P; Raiman, O.; Devès, P. (1987). Troubleshooting: when modelling is the trouble. In Proceedings of AAAI'87, the Sixth National Conference on Artificial Intelligence. Seattle, Washington. Morgan Kaufmann Publishers, Inc. Also in Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 435-440. Morgan Kaufmann: San Mateo, California.

Davis, R.; Shrobe, H.; Hamscher, W.; Wieckert, K.; Shirley, M.; Polit, S. (1982). Diagnosis based on Description of Structure and Function. In Proceedings of *AAAI-82.*

Davis, R. (1983). Diagnosis via causal reasoning: paths of interaction and the locality principle. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 535-541. Morgan Kaufmann: San Mateo, California. Originally in Proceedings of *AAAI-83.*

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence* 24(1), 347-410. Also in *Readings in Model-Based Diagnosis* (1992), Hamscher, W., Console, L. and de Kleer, J. (Eds.), Morgan Kaufmann, 376-407.

Davis, R. (1987). Robustness and transparency in intelligent systems. In *Proceedings of the Third Australian Conference on the Applications of Expert Systems,* 143-154.

Davis, R. (1988). Model-Based Reasoning: Troubleshooting. A.I. Memo No. 1059. Massachusetts Institute of Technology, Artificial Intelligence Laboratory. July 1988.

Davis, R. (1993). Retrospective on "Diagnostic reasoning based on structure and behavior". In Artificial Intelligence 59, 149-157.

de Kleer, J. (1991). Focusing on Probable Diagnoses. In Proceedings of the National Conference on Artificial Intelligence (AAAI), 842-848. Anaheim, July 1991. Morgan Kaufmann Publishers, Inc.

de Kleer, J. & Brown, J. S. (1984). A Qualitative Physics Based on Confluences. *Artificial Intelligence* 24(1-3). Also in Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 88-126. Morgan Kaufmann: San Mateo, California.

de Kleer, J. & Williams, B. C. (1987). Diagnosing Multiple Faults. *Artificial Intelligence* 32, 97-130.

de Kleer, J.; Williams, B. C. (1989). Diagnosis with Behavioral Modes. In Proceedings of *The Eleventh International Joint Conference on Artificial Intelligence*, 1324-1330. Detroit: Morgan Kaufmann Publishers, Inc.

de Kleer, J.; Mackworth, A. K.; Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence* 56, 197-222. Elsevier.

Dormoy, J. L. & Raiman, O. (1988). Assembling a device. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 306-311. Morgan Kaufmann: San Mateo, California.

Downing, K. (1993). Physiological applications of consistency-based diagnosis. *Artificial Intelligence in Medicine Journal* (5), 65-82.

Dressler, O.; Böttcher, C.; Montag, M.; Brinkop, A. (1993). Qualitative and Quantitative Models in a Model-based Diagnosis System for Ballast Tank Systems. In Proceedings of *ToolDiag '93*, Toulouse, France.

Dugdale, J. (1996). Cooperative Problem-solving using Assumption Based Truth Maintenance. In Proceedings of *COOP '96, The Second International Conference on the Design of Cooperative Systems.* Juan-les-Pins, France.

Forbus, K. D. & de Kleer, J. (1993). *Building problem solvers.* MIT Press: Cambridge, Mass.

Forbus, K. D. & Falkenhainer, B. (1990). Self-Explanatory Situations: An integration of qualitative and quantitative knowledge. In Proceedings of *The 8th National Conference on Artificial Intelligence*, Boston, MA, USA.

Forbus, K. D. (1984). Qualitative Process Theory. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 178-219. Morgan Kaufmann: San Mateo, California.

Forbus, K. D. (1988). Qualitative Physics: Past, Present, and Future. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,*.11-39. Morgan Kaufmann: San Mateo, California.

Garrett, R. H. (1991). *Hot and cold water supply.* British Standards Institution (BSI). BSP Professional Books: Oxford.

Genesereth, M. R. (1984). The Use of Design Descriptions in Automated Diagnosis. In Bobrow, D. G. (Ed.) *Qualitative Reasoning about Physical Systems.* North-Holland: Amsterdam.

Jammu, V. B.; Danai, K.; Lewicki, D. G. (1998) Structure-based connectionist network for fault diagnosis of helicopter gearboxes. In *Working papers of the Ninth International Workshop on Principles of Diagnosis*, 70-77. Cape Cod, Massachusetts, USA.

Kuipers, B. (1984). Commonsense Reasoning about Causality: Deriving Behavior from Structure. In Bobrow, D. G. (Ed.) *Qualitative Reasoning about Physical Systems.* North-Holland: Amsterdam.

Kuipers, B. (1986). Qualitative Simulation. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 236-260. Morgan Kaufmann: San Mateo, California.

Lee, M. H.; Ormsby, A. R. T. (1992). Qualitative Modeling of Electrical Circuits. In *QR-92, the Sixth International Workshop about Qualitative Reasoning*, 155-169. Heriot-Watt University, Scotland.

Lind, M. (1994). Modeling Goals and Functions of Complex Industrial Plants. *Applied Artificial Intelligence* 8, 259-283.

McLaughlin, R. K.; McLean, R. C.; Bonthron, W. J. (1981). *Heating Services Design*. Butterworth.

Mavrovouniotis, M. L. & Stephanopoulos, G. (1988). Formal order-of-magnitude reasoning in process engineering. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems*, 323-336. Morgan Kaufmann: San Mateo, California.

Milne, R.; Nicol, C.; Travé-Massuyès, L.; Quevedo, J. (1996). TIGER: knowledge based gas turbine condition monitoring. *AI Communications* 9: 92-108.

Mirzai, A. R. (Ed.). (1990). *Artificial Intelligence : Concepts and Applications in Engineering*. Chapman and Hall: London.

Parsons, S. (1992). Using Interval Algebras to Model Order of Magnitude Reasoning. *Artificial Intelligence in Engineering* 8, 87-98.

Paul, G. (1993). Approaches to abductive reasoning: an overview. *Artificial Intelligence Review* 7, 109-152. Kluwer Academic Publishers.

Poole, D. (1989). Normality and Faults in Logic-Based Diagnosis. In Proceedings of *The Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89*, 1304-1310. Detroit: Morgan Kaufmann Publishers Inc., San Mateo, California.

Price, C. J. (1996). Effortless Incremental Design FMEA. In Proceedings of *The Annual Reliability and Maintainability Symposium, IEEE*, 43-47. Las Vegas.

Price, C. J.; Snooke, N.; Landry, J. (1996). Automated Sneak Identification. *Engineering Applications of Artificial Intelligence* 9(4), 423-427.

Price, C. J. & Taylor, N. S. (1997). Multiple Fault Diagnosis Using FMEA. In Proceedings of *AAAI'97*, 1052-1057.

Raggett, J. & Bains, W. (1992). *Artificial Intelligence from A to Z*.

Raiman, O. (1988). Order of magnitude reasoning. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems*, 318-322. Morgan Kaufmann: San Mateo, California.

Raiman, O. (1991). Order of magnitude reasoning. *Artificial Intelligence* 51, 11-38.

Reggia, J. A.; Nau, D. S.; Wang, P. Y. (1984). Diagnostic expert systems based on a set covering model. In *Developments in Expert Systems*.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence* 32.

Rosenberg, R. & Karnopp, D. (1983). *Introduction to Physical Systems Dynamics*. Mc Graw Hill: New-York.

Rozé, L. (1997). Supervision of Telecommunication Network: A Diagnoser Approach. In working papers of *DX'97, The Eighth International Workshop on Principles of Diagnosis*, 103-111. Le Mont-Saint-Michel, France.

Rozier, D. & Xia, S. (1998). Complex Multiple-Fault Situations: Real-World Circumstances and Advantages of Using MVDS for Their Diagnosis. In working papers of *DX'98, the Ninth International Workshop on Principles of Diagnosis*. Cape Cod, MA, USA.

Rozier, D. & Xia, S. (1997). MVDS: a Strategy for Complex Multiple-Fault Situations Fault-Diagnosis. In working papers of *DX'97, the Eighth International Workshop on Principles of Diagnosis*. Le Mont Saint-Michel, France.

Rozier, D.; Xia, S.; Luker, P. (1997). Organising the Modelling Knowledge for Tailored Predictions in Physical Systems Fault-Diagnosis. In *Proceedings of UKSIM 97, the Conference of the UK Simulation Society*. Keswick, Cumbria, United-Kingdom.

Sachenbacher, M.; Malik, A.; Struss, P. (1998). From electrics to emissions: Experiences in applying model-based diagnosis to real problems in real cars. *In Working Papers of DX'98, The Ninth International Workshop on Principles of Diagnosis*, 246-253. Cape Cod, Massachusetts, USA.

Struss, P. (1987). Problems of interval-based qualitative reasoning. In Weld, D. S.; de Kleer, J. (ed.) *Readings in Qualitative Reasoning about Physical Systems,* 288-305. Morgan Kaufmann: San Mateo, California.

Struss, P. (1988). Mathematical aspects of qualitative reasoning. *Artificial Intelligence in Engineering* 3(3), 156-169. Computational Mechanics Publications.

Struss, P. & Dressler, O. (1989). Physical Negation- Integrating Fault Models into the General Diagnostic Engine. In Proceedings of *The Eleventh International Joint Conference on Artificial Intelligence,* 1318-1323, Detroit. Morgan Kaufmann Publishers, Inc.

Struss, P. (1991). A Theory of Model Simplification and Abstraction for Diagnosis. In Proceedings of *The 5th International Workshop on Qualitative Reasoning, QR-91.*

Struss, P. (1992). What's in SD? Towards a Theory of Modeling for Diagnosis. In Hamscher, W. et al. (Ed.) *Readings in Model-based Diagnosis,* 419-450. Morgan Kaufmann: San Mateo, California.

Struss, P.; Malik, A.; Sachenbacher, M. (1996). Qualitative Modeling is the Key to Automated Diagnosis In *Proceedings of the 13th World Congress of IFAC*, San Francisco, CA, USA, Pergamon

Thoma, J. U. (1975). *Introduction to Bond Graphs and their Applications.* Pergamon Press.

Top, J. L.; Akkermans, J. M.; Breedveld, P. C. (1991). Qualitative Reasoning about Physical Systems: an Artificial Intelligence Perspective. *Journal of the Franklin Institute* 328 (No. 5/6), 1047-1065.

Torasso, P. & Console, L. (1989). *Diagnostic Problem Solving.* North Oxford Academic.

Travé-Massuyès, L & Milne, R. (1995). Application oriented qualitative reasoning. *The Knowledge Engineering Review* 10(2), 181-204.

Travé-Massuyès, L & Milne, R. (1996). Diagnosis of Dynamic Systems Based On Explicit And Implicit Models: An Application to Gas Turbines in Esprit Project TIGER. *Applied Artificial Intelligence Journal* 10(3), 257-277.

Travé-Massuyès, L & Milne, R. (1997). Gas-Turbine Condition Monitoring Using Qualitative Model-Based Diagnosis. In *IEEE Expert*, May/June 1997.

Wick, M. R. (1994). Explanation as a primary task in problem-solving. *The Knowledge Engineering Review* 9(1), 78-82.

Williams, B. C. (1988). MINIMA, A symbolic approach to qualitative algebraic reasoning. In Weld, S. & de Kleer, J. (Ed.) *Readings in Qualitative Reasoning about physical systems*, 312-317. Morgan Kaufmann: San Mateo, California.

Williams, B. C. (1991). A theory of interactions: unifying qualitative and quantitative algebraic reasoning. *Artificial Intelligence* 51, 39-94. Elsevier.

Woods, D. D. 1985. Cognitive Technologies: The Design of Joint Human-Machine Cognitive Systems. *AI Magazine* 4, 86-92.

Xia, S.; Linkens, D. A.; Bennett, S. (1992). Integration of qualitative reasoning and bond graphs: an engineering approach. In Breedveld, P. C. & Dauphin-Tanguy, G. (Ed.) *Journal of IMACS Transactions:* Special issue on Bond graphs for engineers.

Xia, S. (1993). Automatic modelling and analysis of dynamic physical systems using qualitative reasoning and bond graphs, S.Xia, D.A.Linkens, S.Bennett, *in Intelligent Systems Engineering* 2(3).

Appendix A

Listings of MVDS' programs

This appendix A contains the listings of all the programs used to produced results used in this dissertation. They use the language PROLOG for Windows.

The central program MVDS.pl contains no application-dependent information. This information is contained in the file data_heat.pl for the hot-water and heating system, and in the file data_air.pl for the air-conditioning example. The qualitative algebra is declared in the file algebra.pl.

The central program: MVDS.pl

```
%This file contains MVDS' programme itself.
%It does not contain application-dependent information.
%It uses the file data_heat.pl which contains the application-dependent
%information.
%It uses also the file algebra.pl which contains the qualitative algebra.

%---------- open files ----------

:- ensure_loaded(data_heat).
:- ensure_loaded(algebra).

%---------- tools ----------

%overlap(X,Y) is true if the two lists have at least
```

```
%one member in common.

overlap(X,Y):- member(Z,X), member(Z,Y).

%superset(L1, L2) is true if L1 is a superset of L2 (large)
%superset([1],[1]) is false, superset([1,2],[2,1]) is false.

superset(L1, L2):- not subset(L1,L2).
subset(L1, L2):-
member(Y, L2), not member(Y, L1).

%equivalent(L1, L2) is true if L2 contains the same elements as L1
%but in a different order.

equivalent([],[]).
equivalent([X],[X]).
equivalent(A, B):-
        member(X,A),
        member(X,B),
        remove(X,A,A1),
        remove(X,B,B1),
        equivalent(A1,B1).

%List1 and List2 are lists of lists.
%eliminate(List1, List2) is true if List2 is the result of taking out
%of List1 the lists which are supersets of another, or a repeat of
%another.

eliminate([],[]).
eliminate([X|L1], L2):-
        eliminate(L1, [X], L2),!.
eliminate([],L,L ).
eliminate([X|Lavoir], Listsofar, Listfinale):-
        member(Y, Listsofar),
        superset(X, Y),
        eliminate(Lavoir, Listsofar, Listfinale).
eliminate([X|Lavoir], Listsofar, Listfinale):-
        member(Y, Listsofar),
        superset(Y, X),
        remove(Y, Listsofar, Newlistsofar1),
        Newlistsofar2=[X|Newlistsofar1],
```

```
            eliminate(Lavoir, Newlistsofar2, Listfinale).
eliminate([X|Lavoir], Listsofar, Listfinale):-
            Newlistsofar=[X|Listsofar],
            eliminate(Lavoir, Newlistsofar, Listfinale).

%separate(List[of is_a_conf], List2, List3, List4[of var_loc]) is true
%if out of the elements is_a_conf(List, Var, Loc), the elements
%List are compiled in List2, the Var in List3, the Loc in List4.

separate([], [], [], []).
separate([is_a_conf(List, Var, Loc)|Conflist], [List|Sets],
[Var|Vars], [var_loc(Var, Loc)|Quants]):-
            separate(Conflist, Sets, Vars, Quants).

%is_a_subset(Sub, Set) is true if Sub is a subset of Set.

is_a_subset(Sub, Set):-
            member(X, Set),
            append(List1, [X|List2], Set),
            append(List1, List2, Sub).

%describe(List) displays the elements of List, one on each line.

describe([]).
describe([X|List]):-
            write(X),nl,
            describe(List).

%selectvar(List[of value(Val, Var1, Loc, Env), Var2, List2, Rest)
%is true if the values such that Var1 is the same as Var2 are compiled
%in List2, and the others in Rest.

selectvar([], Var, [], []).
selectvar([value(Val, Var, Loc, Env)|L], Var,
[value(Val, Var, Loc, Env)|List], Rest):-
            selectvar(L, Var, List, Rest).
selectvar([value(Val, Var2, Loc, Env)|L], Var1, List,
[value(Val, Var2, Loc, Env)|Rest]):-
            not =(Var1, Var2),
            selectvar(L, Var1, List, Rest).
```

```
same(value(Val, Var, Loc, _), value(Val, Var, Loc, _)).
samequ(var_loc(Var, Loc), var_loc(Var, Loc)).

epure(List, L):-
        member(X1, List),
        remove(X1, List, Rest),
        member(X2, Rest),
        same(X1,X2),
        remove(X1, List, List1),
        epure(List1, L), !.
epure(List, List).

epure2(List, L):-
        member(X1, List),
        remove(X1, List, Rest),
        member(X2, Rest),
        X1=X2,
        remove(X1, List, List1),
        epure2(List1, L), !.
epure2(List, List).
%---------- end of tools ----------

%---------- algorithm ----------

%calculate(Compid, Value1, Value2) is used in the predicate 'complete',
%and is true if Value2 can be obtained from Value1 through model of Compid.

calculate(Compid, value(Val,Var,Loc,Env), Listpred):-
        findall(L, calculate_one_port(Compid, value(Val,Var,Loc,Env), L), Listval),
        findall(value(Val2,Var2,Loc2, Env2), member(value(Val2,Var2,Loc2, Env2), Listval), Listpred).

calculate_one_port(Compid, value(Val,Var,Loc,Env), L):-
        component(Compid, Comp),
        variable(Comp, Var, Newport),
        locate(Loc, Compid, Port1),
        model(Comp, Var, Newport, Relation, Port1),
        rule(Newval, Relation, Val),
        locate(Newloc, Compid, Newport),
        append(Env, [Compid], Newenv),
        L=value(Newval, Var, Newloc, Newenv),!.
```

```
calculate_one_port(Compid, value(Val,Var,Loc,Env), L):-
        component(Compid, Comp),
        variable(Comp, Var, Newport),
        locate(Loc, Compid, Port1),
        model(Comp, Var, Port1, Relation, Newport),
        rule(Val, Relation, Newval),
        locate(Newloc, Compid, Newport),
        append(Env, [Compid], Newenv),
        L=value(Newval, Var, Newloc, Newenv),!.
calculate_one_port(Compid, value(Val, Var, Loc, Env),L):-
        component(Compid, Comp),
        variable(Comp, Var, Newport),
        locate(Loc, Compid, Port1),
        model(Comp, Var, Newport, Relation, Port1),
        not rule(Newval, Relation, Val),
        Newenv=[Compid|Env],
        L= is_a_conf(Newenv, Var, Loc).

%system(List) is true if List compiles the components declared in the file DATA.PL

system(List):- findall(Compo, component(Compo, _), List).

contrad_one(value(Val1, Var, Loc, Env1), [], []):-!.
contrad_one(value(Val1, Var, Loc, Env1), [value(Val2, Var, Loc, Env2)|Listvalue], Listcontrad):-
        Val1=Val2,
        contrad_one(value(Val1, Var, Loc, Env1), Listvalue, Listcontrad).
contrad_one(value(Val1, Var, Loc, Env1), [value(Val2, Var, Loc, Env2)|Listvalue], [value(Val2, Var, Loc, Env2)|Listcontrad]):-
        contrad_one(value(Val1, Var, Loc, Env1), Listvalue, Listcontrad).

%conflict(Env, Obs, Conf) is true if spreading the values in Obs through
%the models of the components in Env gives, amongst others, the conflict set Conf.

conflicts(List_of_values, X, X):-length(List_of_values, 1), !.
conflicts([value(Val1, Var, Loc, Env1)|List_of_values], Conflict_list_sofar, Final_list):-
        findall(is_a_conf(Conf, Var, Loc), contradiction(value(Val1, Var, Loc, Env1), List_of_values, Conf), Part_list),
        append(Conflict_list_sofar, Part_list, New_list_sofar),
        conflicts(List_of_values, New_list_sofar, Final_list).

contradiction(value(Val1, Var, Loc, Env1), List, Conf):-
        member(value(Val2, Var, Loc, Env2), List),
        not(Val1=Val2),
```

```
        append(Env1, Env2, Conf1),
        epure2(Conf1, Conf),
        nl, write('Symptom at loc. '), write(Loc), write(' : '),
        write(Var), write(' = '), write(Val1), write(' vs '), write(Var),
        write(' = '), write(Val2),nl, write(' (resulting conflict set: '),
        write(Conf), write(' )').

%relevant_comp(Listcomp, Comp, value(Val, Var, Loc, Env)) is true if
%the component Comp, from the list Listcomp, has a port on location Loc,
%is not included in the environment Env.

relevant_comp(Listcomp, Comp, value(Val, Var, Loc, Env)):-
        member(Comp, Listcomp),
        not member(Comp, Env),
        locate(Loc, Comp, _).

%propagate_value(value(Val, Var, Loc, Env), List_relev_comp, Listnewpreds)
%is true if propagating the value(Val, Var, Loc, Env) through the components
%in List_relev_comp give the predictions contained in Listnewpreds.

propagate_value(value(Val, Var, Loc, Env), [Comp|List_relev_comp], Listpreds):-
        propagate_value(value(Val, Var, Loc, Env), [Comp|List_relev_comp], [], Listpreds).
propagate_value(_, [], V,V):- !.
propagate_value(value(Val, Var, Loc, Env), [Comp|List_relev_comp],
                Sofarpreds, Listpreds):-
        calculate(Comp, value(Val, Var, Loc, Env), Listval),
        append(Sofarpreds, Listval, Newsofarpreds),
        propagate_value(value(Val, Var, Loc, Env), List_relev_comp, Newsofarpreds, Listpreds).

%predictions(Listcomp, Listval, Sofarpreds, List)
%is true if List contains the predictions that are obtained from the quantities
%in Listval through the components in Listcomp.
%Sofarpreds is a temporary list of predictions.

predictions(Environ, [], P,P):- !.
predictions(Listcomp,[value(Val, Var, Loc, Env)|Listval], Sofarpreds, L):-
        findall(Comp, relevant_comp(Listcomp, Comp, value(Val, Var, Loc, Env)), List_relev_comp),
        propagate_value(value(Val, Var, Loc, Env), List_relev_comp, Listnewpreds),
        append(Listval, Listnewpreds, Newlist),
        append(Sofarpreds, Listnewpreds,Totalpreds),
        predictions(Listcomp, Newlist, Totalpreds, L),!.
```

```
predictions(Listcomp,[value(Val, Var, Loc, Env)|Listval], Sofarpreds, L):-
        predictions(Listcomp,Listval, Sofarpreds, L).

% candidates(BasevarL) is the central predicate.
% It produces the whole process from BasevarL,
% which contains the base-entities.

candidates(BasevarL) :-
        system(System),
        relevant_entities(Avail),
        candidates(System,Avail, BasevarL, [], BasevarL, []).
candidates(System,Avail,FocusvarL, Predicted, Totalfocus, Pastconf):-
        read_measur(FocusvarL, _, Usef_observ), nl,
        write('------- Predictions obtained about entity '),
        write(FocusvarL), write(' ----'), nl,nl,
        predictions(System,Usef_observ, [], Listpred),
        describe(Listpred),
        append(Usef_observ, Listpred, Forconf),
        nl,write('--------- New symptoms and conflict sets ---------'), nl,
        conflicts(Forconf, [], Newconflist),
        separate(Newconflist, Conf_list2, Vars1, Quants1),
        append(Conf_list2, Pastconf, Totalconf),
        epure(Quants1, Quants2),
        epure2(Vars1, Vars2),
        eliminate(Totalconf, Minconf),
        nl,nl,
        write('------ Updated minimal conflict sets -------'),
        nl,
        describe(Minconf), nl,
        cand2([], Minconf, List),
        write('--------- Updated minimal candidates ----------'),
        nl,
        describe(List), nl,
        write('***** Result assessment *****'), nl,
        further(Avail, System, FocusvarL, Vars2, Quants2, List,
                Listpred, Totalfocus, Conf_list2, Totalconf).

%selectvarL(List[of value], List2[of Var], List3[of value]) is true if
%List3 contains the values from List which are about a variable listed in List2.

selectvarL([], _, []) :- !.
```

```
selectvarL([value(Val, Var, Loc, Env)|Avail], BasevarL,
                    [value(Val, Var, Loc, Env)|Observ]) :-
        member(Var, BasevarL),
        selectvarL(Avail, BasevarL, Observ).
selectvarL([value(Val, Var, Loc, Env)|Avail], BasevarL, Observ) :-
        not member(Var, BasevarL),
        selectvarL(Avail, BasevarL, Observ).

%further analyses all these data and
%is wrong if no more revision is needed: the process stops,
%is true if revision is needed, and starts a new <candidates> function
%with the new entities under focus.

further(Avail, System, FocusvarL, Vars, Quants, List,
        Listpred, Totalfocus, Conf_list2, Totalconf) :-
        satisfactory(Avail, Totalfocus),
        write('*** End of result assessment ***'), nl,
        write('*** End of process ***'), nl, !.
further(Avail, System, FocusvarL, Vars, Quants, List,
        Listpred, Totalfocus, Conf_list2, Totalconf) :-
        =(Conf_list2, []),
        write('Last diagnosis session revealed no new conflicts => End'),
        nl, write('*** End of result assessment ***'),
        nl, write('*** End of process ***'), !.
further(Avail, System, FocusvarL, Vars, Quants, List,
        Listpred, Totalfocus, Conf_list2, Totalconf) :-
        write('> Global influences: '),
        modelling_global(Avail, Totalfocus, Vars, Newfocus1),
        write('> Local influences: '),
        modelling_local(Avail, Totalfocus, Quants, Newfocus2),
        append(Newfocus1, Newfocus2, Newfocus3),
        not =([], Newfocus3),
        write('*** End of result assessment and start of dynamic revision***'),nl,
        revision(Newfocus3,Avail,Totalfocus, System, Listpred, Totalconf, List),!.
further(Avail, System, FocusvarL, Vars, Quants, List,
        Listpred, Totalfocus, Conf_list2, Totalconf) :-
        write('The entities involved in new conflicts are not causally linked
        => End'),
        nl, write('*** End of result assessment ***'),
        nl, write('*** End of process ***'), !.
```

```
%satisfactory(Avail, FocusvarL) is true if
%the entities in Avail are the same as in FocusvarL

satisfactory(Avail, FocusvarL) :-
        equivalent(FocusvarL, Avail),
        write('All available entities are already under focus.'), nl.

%the first part of revision(Newfocus3,Avail,Totalfocus, System, Listpred, Totalconf,List)
%is true if NewfocusvarL is the union of Newfocus3 and Totalfocus,
%and if the process is not terminated by the operator.
%The second part of the predicate is true if there is termination
%by the operator.

revision(Newfocus3,Avail,Totalfocus, System, Listpred, Totalconf,List):-
        epure2(Newfocus3, Newfocus),
        append(Newfocus, Totalfocus, NewfocusvarL), nl,
        write('The new entities under focus are '), write(Newfocus),
        nl, write('*** End of dynamic revision ***'),
        not terminate,
        nl, write('*** Start of further diagnostic reasoning ***'),nl,
        candidates(System,Avail, Newfocus,Listpred, NewfocusvarL, Totalconf).
revision(Newfocus3,Avail,Totalfocus, System, Listpred, Totalconf, List):-
        nl,write('*** Process terminated by the operator ***'),
        nl,write('Final minimal candidates:'),
        nl,describe(List), !.

%terminate is true if the operator types 'end' when prompted.

terminate:-
        nl,nl,
        write('To terminate type [end], to go further type [go]'),nl,
        write('followed by a dot and press the [enter] key.'),
        read(Order),
        Order = end.

%read_measur(List, Measursofar, MeasurL) is true if
%the operator types the measurements in MeasurL
%about the entities in List.

read_measur([], Measursofar, Measursofar).
```

```
read_measur([Entity|Newfocus], Measursofar, MeasurL):-
        nl,
        write('Enter the measurements for entity '), write(Entity),
        write(', with format:'), nl,
        write('[value(qualitat. value, '),write(Entity),
        write(', location, []), value(...), ...].'),
        nl,nl, write('Measurements '), read(List),
        append(List, Measursofar, Newmeasursofar),
        read_measur(Newfocus, Newmeasursofar, MeasurL).

%modelling_global(Avail, FocusvarL, Vars, Newfocus) is true if
%the entities in FocusvarL are globally linked with the entities
%in Newfocus (which were not already under focus).
%It uses the predicate modelling_glob below.

modelling_global(Avail, FocusvarL, Vars, Newfocus) :-
        global_infl(_,_),
        modelling_glob(Avail, Avail, FocusvarL, Vars, Newfocus),!.
modelling_global(Avail, FocusvarL, Vars, []) :-
        nl, write('No global influence has been declared.'), nl.

%modelling_local(Avail, FocusvarL, Quants, Newfocus) is true if
%the quantities in Quants are locally linked with quantities of which
% entities are the list Newfocus (entities not already under focus).
%It uses the predicate modelling_loc below.

modelling_local(Avail, FocusvarL, Quants, Newfocus) :-
        local_infl(_,_,_),
        modelling_loc(Avail, Avail, FocusvarL, Quants, Newfocus),!.
modelling_local(Avail, FocusvarL, Quants, []):-
        nl, write('No local influence has been declared.'), nl.

modelling_loc(_, [], _, _, []).
modelling_loc(Firstavail, Avail, FocusvarL, Quants, [Var2|Newfocus]) :-
        member(Var2, Avail),
        local_infl(Compid, Var1, Var2),
        locate(Loc, Compid, _),
        member(var_loc(Var1, Loc), Quants),
        not member(Var2, FocusvarL),
        remove(Var2, Avail, Newavail),
        nl, write('Entity <'), write(Var2),
```

```
            write('> put under focus because local influence'), nl,
            write('between <'),write(Var2), write('> and <'), write(Var1),
            write('> (involved in conflicts),'),
            nl, write('within component '), write(Loc), nl,
            modelling_loc(Firstavail, Newavail, FocusvarL, Quants, Newfocus).
modelling_loc(Firstavail, Avail, FocusvarL, Quants, [Var2|Newfocus]) :-
            member(Var2, Avail),
            local_infl(Compid, Var2, Var1),
            locate(Loc, Compid, _),
            member(var_loc(Var1, Loc), Quants),
            not member(Var2, FocusvarL),
            remove(Var2, Avail, Newavail),
            nl, write('Entity <'), write(Var2),
            write('> put under focus because local influence'), nl,
            write('between <'),write(Var2), write('> and <'), write(Var1),
            write('> (involved in conflicts),'),
            nl, write('within component '), write(Compid), nl,
            modelling_loc(Firstavail, Newavail, FocusvarL, Quants, Newfocus).
modelling_loc(Firstavail, Avail, FocusvarL, Quants, []):-
            Avail=Firstavail,
            nl, write('No local influence of interest.'), nl, !.
modelling_loc(Firstavail, Avail, FocusvarL, Quants, []):- !.

modelling_glob(_, [], _, _, []).
modelling_glob(Firstavail, Avail, FocusvarL, Vars, [Var2|Newfocus]) :-
            member(Var2, Avail),
            global_infl(Var1, Var2),
            member(Var1, Vars),
            not member(Var2, FocusvarL),
            remove(Var2, Avail, Newavail),
            nl, write('Entity <'), write(Var2),
            write('> put under focus because global influence'), nl,
            write('between <'),
            write(Var2), write('> and <'), write(Var1),
            write('> (involved in conflicts)'), nl,
            modelling_glob(Firstavail, Newavail, FocusvarL, Vars, Newfocus).
modelling_glob(Firstavail, Avail, FocusvarL, Vars, [Var2|Newfocus]) :-
            member(Var2, Avail),
            global_infl(Var2, Var1),
            member(Var1, Vars),
            not member(Var2, FocusvarL),
```

```
        remove(Var2, Avail, Newavail),
        nl, write('Entity <'), write(Var2),
        write('> put under focus because global influence'), nl,
        write('between <'),
        write(Var2), write('> and <'), write(Var1), write('> (involved in conflicts)'), nl,
        modelling_glob(Firstavail, Newavail, FocusvarL, Vars, Newfocus).
modelling_glob(Firstavail, Avail, FocusvarL, Vars, []) :-
        Avail=Firstavail,
        nl, write('No global influence of interest.'), nl, !.
modelling_glob(Firstavail, Avail, FocusvarL, Vars, []) :- !.

% cand2 gives the minimal candidate sets regarding several new conflicts

cand2(Locand, [], []).
cand2(Locand, [X|[]], Lmin):-
        findall( Cand, cand(Locand, X, Cand), List),
        eliminate(List, Lmin).
cand2(Locand, [X|[Y|Loconf]], L):-
        findall( Cand, cand(Locand, X, Cand), List),
        eliminate(List, Lmin),
        cand2(Lmin,[Y|Loconf], L).

%cand(List1, List2, List3) is true if the list of candidates in List1
%turns into the list of candidates in List3 when the conflict set List2
% is discovered.

cand([], Conf, [X]):- member(X, Conf).
cand(Oldcandlist, Conf, Cand):-
        member(Cand, Oldcandlist),
        overlap(Cand, Conf).
cand(Oldcandlist, Conf, Cand):-
        member(X, Oldcandlist),
        not overlap(X, Conf),
        member(Y, Conf),
        append(X, [Y], Cand).
```

Models and structure for the air-conditioning system: data_air.pl

```
%This file contains the library of models and
% the physical system's structure
%for the air-conditioning example.

%-----------------------------------------
%system's description

%relevant_entities([temp, flow, dust, water]).
relevant_entities([dust]).

component(f1, airfilter).
component(f2, airfilter).
component(h, airheater).

locate(p1, f1, in).
locate(p2, f1, out).
locate(p2, h, in).
locate(p3, h, out).
locate(p3, f2, in).
locate(p4, f2, out).

local_infl(h, dust, temp).
local_infl(h, flow, water).

global_infl(flow, dust).

%-----------------------------------------
% library

%-----------------------------------
%air filter with inlet <in> and outlet <out>

variable(airfilter, flow, in).
variable(airfilter, flow, out).
model(airfilter, flow, out, ==, in).

variable(airfilter, dust, in).
variable(airfilter, dust, out).
```

```
model(airfilter, dust, out, <<, in).

variable(airfilter, water, in).
variable(airfilter, water, out ).
model(airfilter, water, out, <<, in).

variable(airfilter, temp, in).
variable(airfilter, temp, out ).
model(airfilter, temp, out, ==, in).

%-------------------------------
%air heater with inlet <in> and outlet <out>

variable(airheater, flow, in).
variable(airheater, flow, out).
model(airheater, flow, out, ==, in).

variable(airheater, dust, in).
variable(airheater, dust, out).
model(airheater, dust, out, ~>, in).

variable(airheater, water, in).
variable(airheater, water, out).
model(airheater, water, out, ~>, in).

variable(airheater, temp, in).
variable(airheater, temp, out).
model(airheater, temp, out, >>, in).
```

Models and structure for the hot-water and heating system: data_heat.pl

```
%This file contains the library of models and
% the physical system's structure
%for the hot-water and heating system.

%-----------------------------------------
%Physical system's structure

relevant_entities([temp, flow, pressure, air, hard]).
```

```
component(b2, boilerfour).
component(r1, radiator).
component(r2, radiator).
component(p, waterpump).
component(v, vessel).
component(s, softener).

locate(p1, s, in).
locate(p2, s, out).
locate(p2, v, in2).
locate(p3, v, out2).
locate(p4, v, out1).
locate(p4, b2, in1).
locate(p5, b2, out1).
locate(p5, p, in).
locate(p6, p, out).
locate(p6, r1, in).
locate(p7, r1, out).
locate(p7, r2, in).
locate(p8, r2, out).
locate(p8, b2, in2).
locate(p9, b2, out2).
locate(p9, v, in1).

%local_infl(_, _, _):- false.
local_infl(r1, air, temp).
local_infl(r2, air, temp).
local_infl(r1, flow, temp).
local_infl(r2, flow, temp).
local_infl(v, flow, temp).

%global_infl(_, _):- false.
global_infl(air, hard).
global_infl(hard, flow).
%global_infl(pressure, flow).

%-----------------------------------
% Library of models
%-----------------------------------
```

```
%-----------------------------------------
%water boiler with inlet <in> and outlet <out>

variable(boiler, flow, in).
variable(boiler, flow, out).
model(boiler, flow, in, ==, out).

variable(boiler, pressure, in).
variable(boiler, pressure, out).
model(boiler, pressure, in,==, out).


variable(boiler, temp, in).
variable(boiler, temp, out).
model(boiler, temp, out, >>, in).

variable(boiler, hard, in).
variable(boiler, hard, out).
model(boiler, hard, in, ==, out).

variable(boiler, air, in).
variable(boiler, air, out).
model(boiler, air, in,==, out).

%-----------------------------------------
%water softener with inlet <in> and outlet <out>

variable(softener, flow, in).
variable(softener, flow, out).
model(softener, flow, in, ==, out).

variable(softener, pressure, in).
variable(softener, pressure, out).
model(softener, pressure, in,==, out).

variable(softener, temp, in).
variable(softener, temp, out).
model(softener, temp, out, ==, in).

variable(softener, hard, in).
variable(softener, hard, out).
```

```
model(softener, hard, in, >>, out).

variable(softener, air, in).
variable(softener, air, out).
model(softener, air, in,==, out).

%-----------------------------------------
%water boiler <boilerfour> with inlets <in1>and <in2>,
% and outlets <out1> and <out2>.

variable(boilerfour, flow, in1).
variable(boilerfour, flow, out1).
variable(boilerfour, flow, in2).
variable(boilerfour, flow, out2).
model(boilerfour, flow, in1, ==, out1).
model(boilerfour, flow, in2, ==, out2).

variable(boilerfour, pressure, in1).
variable(boilerfour, pressure, out1).
variable(boilerfour, pressure, in2).
variable(boilerfour, pressure, out2).
model(boilerfour, pressure, in1,==, out1).
model(boilerfour, pressure, out2,==, in2).

variable(boilerfour, temp, in1).
variable(boilerfour, temp, out1).
variable(boilerfour, temp, in2).
variable(boilerfour, temp, out2).
model(boilerfour, temp, out1, >>, in1).
model(boilerfour, temp, out2, >>, in2).

variable(boilerfour, hard, in1).
variable(boilerfour, hard, out1).
variable(boilerfour, hard, in2).
variable(boilerfour, hard, out2).
model(boilerfour, hard, in1, ==, out1).
model(boilerfour, hard, in2, ==, out2).

variable(boilerfour, air, in1).
variable(boilerfour, air, out1).
variable(boilerfour, air, in2).
```

```
variable(boilerfour, air, out2).
model(boilerfour, air, in1,==, out1).
model(boilerfour, air, in2,==, out2).

%-----------------------------------------
%water vessel <vessel> with inlets <in1>and <in2>,
%and outlets <out1> and <out2>.
%The ports with suffix <1> relate to the flow from the boiler,
%and the ports with suffix <2> relate to the flow of water stored.

variable(vessel, flow, in1).
variable(vessel, flow, out1).
variable(vessel, flow, in2).
variable(vessel, flow, out2).
model(vessel, flow, in1, ==, out1).
model(vessel, flow, in2, ==, out2).

variable(vessel, pressure, in1).
variable(vessel, pressure, out1).
variable(vessel, pressure, in2).
variable(vessel, pressure, out2).
model(vessel, pressure, in1,==, out1).
model(vessel, pressure, in2,==, out2).

variable(vessel, temp, in1).
variable(vessel, temp, out1).
variable(vessel, temp, in2).
variable(vessel, temp, out2).
model(vessel, temp, out1, <<, in1).
model(vessel, temp, out2, >>, in2).

variable(vessel, hard, in1).
variable(vessel, hard, out1).
variable(vessel, hard, in2).
variable(vessel, hard, out2).
model(vessel, hard, in1, ==, out1).
model(vessel, hard, in2, ==, out2).

variable(vessel, air, in1).
variable(vessel, air, out1).
variable(vessel, air, in2).
```

```
variable(vessel, air, out2).
model(vessel, air, in1,==, out1).
model(vessel, air, in2,==, out2).

%-----------------------------------------
%water waterpump with inlet <in> and outlet <out>

variable(waterpump, flow, in).
variable(waterpump, flow, out).
model(waterpump, flow, in,==, out).

variable(waterpump, pressure, in).
variable(waterpump, pressure, out).
model(waterpump, pressure, out, >>, in).

variable(waterpump, temp, in).
variable(waterpump, temp, out).
model(waterpump, temp, in,==, out).

variable(waterpump, hard, in).
variable(waterpump, hard, out).
model(waterpump, hard, in, ==, out).

variable(waterpump, air, in).
variable(waterpump, air, out).
model(waterpump, air, in, ==, out).

%-----------------------------------------
%water radiator with inlet <in> and outlet <out>

variable(radiator, flow, in).
variable(radiator, flow, out).
model(radiator, flow, in, ==, out).

variable(radiator, pressure, in).
variable(radiator, pressure, out).
model(radiator, pressure, in, ==, out).

variable(radiator, temp, in).
variable(radiator, temp, out).
model(radiator, temp, in,~>, out).
```

```
variable(radiator, hard, in).
variable(radiator, hard, out).
model(radiator, hard, in,==, out).

variable(radiator, air, in).
variable(radiator, air, out).
model(radiator, air, in, ==, out).
```

Qualitative algebra: algebra.pl

```
%This file contains the qualitative algebra.

rule(very_low,<<,medium).
rule(low,<<,high).
rule(medium,<<,very_high).

rule(very_low, ~<,low).
rule(low, ~<,medium).
rule(medium, ~<, high).
rule(high, ~<, very_high).

rule(L, ==, L).

rule(very_high, ~>, high).
rule(high, ~>, medium).
rule(medium, ~>, low).
rule(low, ~>, very_low).

rule(very_high, >>, medium).
rule(high,>>,low).
rule(medium,>>,very_low).
```

Appendix B

Outputs of MVDS' processes on the case-study

This appendix B contains a series of outputs produced by MVDS on the hot-water and heating system. They complete the output printed in chapter 5. The conditions of the diagnostic process that has produced each output are described in the titles.

Scenario 1, rich set of measurements

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...),
...].

Measurements |: [value(very_high, temp, p6, []),
value(very_high, temp, p7, []), value(very_high, temp,
p9, []), value(low, temp, p1, [])].

------ Predictions obtained about entity [temp] -----

value(high,temp,p7,[r1])
value(very_high,temp,p5,[p])
value(high,temp,p8,[r2])
value(medium,temp,p8,[b2])
value(medium,temp,p4,[v])
value(low,temp,p2,[s])
value(medium,temp,p8,[r1,r2])
value(medium,temp,p4,[p,b2])
value(high,temp,p7,[b2,r2])
value(very_high,temp,p5,[v,b2])
value(high,temp,p3,[s,v])
value(very_high,temp,p9,[r1,r2,b2])
value(very_high,temp,p9,[p,b2,v])
value(very_high,temp,p6,[b2,r2,r1])
value(very_high,temp,p6,[v,b2,p])
value(medium,temp,p4,[r1,r2,b2,v])
value(very_high,temp,p5,[b2,r2,r1,p])
value(high,temp,p7,[v,b2,p,r1])
value(medium,temp,p8,[v,b2,p,r1,r2])

---------- New symptoms and conflict sets ----------

Symptom at loc. p7 : temp = very_high vs temp = high
 (resulting conflict set: [r1] )
Symptom at loc. p7 : temp = very_high vs temp = high
 (resulting conflict set: [b2,r2] )
```

Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [v,b2,p,r1])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [r2,b2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [v,b2,p,r1,r2])

------- Updated minimal conflict sets -------
[b2,r2]
[r1]

--------- Updated minimal candidates ----------
[b2,r1]
[r2,r1]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
Entity <flow> put under focus because local influence between <flow> and <temp> (involved in conflicts), within component r1

Entity <air> put under focus because local influence between <air> and <temp> (involved in conflicts), within component r1
*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]
*** End of dynamic revision ***

To terminate type [end], to go further type [go] followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p1, []), value(high, flow, p5, []), value(high, flow, p6, []), value(high, flow, p8, [])].

Enter the measurements for entity air, with format:
[value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(low, air, p2, []), value(low, air, p6, []), value(medium, air, p7, []), value(medium, air, p8, [])].

------- Predictions obtained about entity [flow,air] ----

value(low,air,p3,[v])
value(low,air,p1,[s])
value(low,air,p7,[r1])
value(low,air,p5,[p])
value(medium,air,p6,[r1])
value(medium,air,p8,[r2])
value(medium,air,p9,[b2])
value(medium,air,p7,[r2])
value(high,flow,p2,[s])
value(high,flow,p4,[b2])
value(high,flow,p6,[p])
value(high,flow,p7,[r1])
value(high,flow,p5,[p])
value(high,flow,p9,[b2])
value(high,flow,p7,[r2])
value(low,air,p8,[r1,r2])
value(low,air,p4,[p,b2])
value(medium,air,p5,[r1,p])
value(medium,air,p9,[r2,b2])
value(medium,air,p4,[b2,v])
value(medium,air,p6,[r2,r1])
value(high,flow,p3,[s,v])
value(high,flow,p9,[b2,v])
value(high,flow,p7,[p,r1])
value(high,flow,p8,[r1,r2])
value(high,flow,p4,[p,b2])
value(high,flow,p4,[b2,v])
value(high,flow,p6,[r2,r1])

value(low,air,p9,[r1,r2,b2])
value(low,air,p9,[p,b2,v])
value(medium,air,p4,[r1,p,b2])
value(medium,air,p4,[r2,b2,v])
value(medium,air,p5,[r2,r1,p])
value(high,flow,p8,[p,r1,r2])
value(high,flow,p9,[r1,r2,b2])
value(high,flow,p9,[p,b2,v])
value(high,flow,p5,[r2,r1,p])
value(low,air,p4,[r1,r2,b2,v])
value(medium,air,p9,[r1,p,b2,v])
value(medium,air,p4,[r2,r1,p,b2])
value(high,flow,p9,[p,r1,r2,b2])
value(high,flow,p4,[r1,r2,b2,v])
value(high,flow,p4,[r2,r1,p,b2])
value(medium,air,p9,[r2,r1,p,b2,v])
value(high,flow,p4,[p,r1,r2,b2,v])
value(high,flow,p9,[r2,r1,p,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p6 : air = low vs air = medium
(resulting conflict set: [r1])
Symptom at loc. p6 : air = low vs air = medium
(resulting conflict set: [r2,r1])
Symptom at loc. p7 : air = medium vs air = low
(resulting conflict set: [r1])
Symptom at loc. p8 : air = medium vs air = low
(resulting conflict set: [r1,r2])
Symptom at loc. p7 : air = low vs air = medium
(resulting conflict set: [r1,r2])
Symptom at loc. p5 : air = low vs air = medium
(resulting conflict set: [r1,p])
Symptom at loc. p5 : air = low vs air = medium
(resulting conflict set: [r2,r1,p])
Symptom at loc. p8 : air = medium vs air = low
(resulting conflict set: [r1,r2])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [p,b2,v])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [p,b2,v])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [p,r2,b2,v])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [r2,p,b2,v])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r1,p,b2,v])
Symptom at loc. p9 : air = low vs air = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [r1,r2,b2,v])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [v,r2,r1,p,b2])

------- Updated minimal conflict sets -------
[b2,r2]
[p,b2,v]
[r1]

---------- Updated minimal candidates ----------
[b2,r1]
[r2,p,r1]
[r2,v,r1]

**** Result assessment ****
> Global influences:

Entity <hard> put under focus because global influence
between <hard> and <air> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go] followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format: [value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(low, hard, p8, []), value(very_low, hard, p4, []), value(very_low, hard, p6, []), value(very_low, hard, p9, [])].
Measurements |: [value(low, hard, p8, []), value(very_low, hard, p4, []), value(very_low, hard, p6, []), value(very_low, hard, p9, [])].

------ Predictions obtained about entity [hard] -----

value(low,hard,p9,[b2])
value(low,hard,p7,[r2])
value(very_low,hard,p5,[b2])
value(very_low,hard,p9,[v])
value(very_low,hard,p7,[r1])
value(very_low,hard,p5,[p])
value(very_low,hard,p8,[b2])
value(very_low,hard,p4,[v])
value(low,hard,p4,[b2,v])
value(low,hard,p6,[r2,r1])
value(very_low,hard,p6,[b2,p])
value(very_low,hard,p8,[v,b2])
value(very_low,hard,p8,[r1,r2])
value(very_low,hard,p4,[p,b2])
value(very_low,hard,p7,[b2,r2])
value(very_low,hard,p5,[v,b2])
value(low,hard,p5,[r2,r1,p])
value(very_low,hard,p7,[b2,p,r1])
value(very_low,hard,p7,[v,b2,r2])
value(very_low,hard,p9,[r1,r2,b2])
value(very_low,hard,p9,[p,b2,v])
value(very_low,hard,p6,[b2,r2,r1])
value(very_low,hard,p6,[v,b2,p])
value(low,hard,p4,[r2,r1,p,b2])
value(very_low,hard,p8,[b2,p,r1,r2])
value(very_low,hard,p6,[v,b2,r2,r1])
value(very_low,hard,p4,[r1,r2,b2,v])
value(very_low,hard,p5,[b2,r2,r1,p])
value(very_low,hard,p7,[v,b2,p,r1])
value(low,hard,p9,[r2,r1,p,b2,v])
value(very_low,hard,p5,[v,b2,r2,r1,p])
value(very_low,hard,p8,[v,b2,p,r1,r2])

---------- New symptoms and conflict sets ----------

Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [b2])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [v,b2])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p,r1,r2])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [b2,v])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p6 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [b2])
Symptom at loc. p9 : hard = very_low vs hard = low

(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [b2,v])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [p,b2,v])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r2,b2,p,r1])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r2,v,b2,p,r1])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [b2,r2,r1,p])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [b2,v])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [v,r2,r1,p,b2])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [v,p,b2])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r1,r2,b2,v])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,b2,p])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,v,b2,p])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2,r1])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1,p])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1,p])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2,r1,p])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [p,r1,r2,b2,v])

-------- Updated minimal conflict sets --------
[r1]
[b2]

---------- Updated minimal candidates ----------
[r1,b2]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.
The entities involved in new conflicts are not causally linked
=> End
*** End of result assessment ***
*** End of process ***

Scenario 1, poor set of measurements

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...),
...].

Measurements |: [value(very_high, temp, p5, []),
value(high, temp, p8, [])].

------ Predictions obtained about entity [temp] -----

value(medium,temp,p4,[b2])
value(very_high,temp,p6,[p])
value(very_high,temp,p7,[r2])
value(very_high,temp,p9,[b2,v])
value(high,temp,p7,[p,r1])
value(medium,temp,p8,[p,r1,r2])
value(very_high,temp,p9,[p,r1,r2,b2])
value(medium,temp,p4,[p,r1,r2,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p8 : temp = high vs temp = medium
 (resulting conflict set: [p,r1,r2] )
Symptom at loc. p7 : temp = very_high vs temp =
high
 (resulting conflict set: [r2,p,r1] )

------- Updated minimal conflict sets --------
[p,r1,r2]

---------- Updated minimal candidates ----------
[r2]
[r1]
[p]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
Entity <flow> put under focus because local influence
between <flow> and <temp> (involved in conflicts),
within component r1

Entity <air> put under focus because local influence
between <air> and <temp> (involved in conflicts),
within component r1
*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p4, []), value(high, flow, p9, [])].

Enter the measurements for entity air, with format:
[value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(low, air, p5, []), value(medium, air, p9, [])].
Measurements |: [value(low, air, p5, []), value(medium, air, p9, [])].

------- Predictions obtained about entity [flow,air] -----

value(low,air,p4,[b2])
value(low,air,p6,[p])
value(medium,air,p8,[b2])
value(medium,air,p4,[v])
value(high,flow,p5,[b2])
value(high,flow,p9,[v])
```

value(high,flow,p8,[b2])
value(high,flow,p4,[v])
value(low,air,p9,[b2,v])
value(low,air,p7,[p,r1])
value(medium,air,p7,[b2,r2])
value(medium,air,p5,[v,b2])
value(high,flow,p6,[b2,p])
value(high,flow,p8,[v,b2])
value(high,flow,p7,[b2,r2])
value(high,flow,p5,[v,b2])
value(low,air,p8,[p,r1,r2])
value(medium,air,p6,[b2,r2,r1])
value(medium,air,p6,[v,b2,p])
value(high,flow,p7,[b2,p,r1])
value(high,flow,p7,[v,b2,r2])
value(high,flow,p6,[b2,r2,r1])
value(high,flow,p6,[v,b2,p])
value(low,air,p9,[p,r1,r2,b2])
value(medium,air,p5,[b2,r2,r1,p])
value(medium,air,p7,[v,b2,p,r1])
value(high,flow,p8,[b2,p,r1,r2])
value(high,flow,p6,[v,b2,r2,r1])
value(high,flow,p5,[b2,r2,r1,p])
value(high,flow,p7,[v,b2,p,r1])
value(low,air,p4,[p,r1,r2,b2,v])
value(medium,air,p8,[v,b2,p,r1,r2])
value(high,flow,p5,[v,b2,r2,r1,p])
value(high,flow,p8,[v,b2,p,r1,r2])

---------- New symptoms and conflict sets ----------

Symptom at loc. p5 : air = low vs air = medium
(resulting conflict set: [v,b2])
Symptom at loc. p5 : air = low vs air = medium
(resulting conflict set: [b2,r2,r1,p])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [b2,v])
Symptom at loc. p9 : air = medium vs air = low
(resulting conflict set: [p,r1,r2,b2])
Symptom at loc. p4 : air = low vs air = medium
(resulting conflict set: [b2,v])
Symptom at loc. p6 : air = low vs air = medium
(resulting conflict set: [p,b2,r2,r1])
Symptom at loc. p6 : air = low vs air = medium
(resulting conflict set: [v,b2,p])
Symptom at loc. p8 : air = medium vs air = low
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p4 : air = medium vs air = low
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p7 : air = low vs air = medium
(resulting conflict set: [p,r1,b2,r2])
Symptom at loc. p7 : air = low vs air = medium
(resulting conflict set: [v,b2,p,r1])
Symptom at loc. p8 : air = low vs air = medium
(resulting conflict set: [v,b2,p,r1,r2])

------- Updated minimal conflict sets --------
[p,r1,r2]
[v,b2]

---------- Updated minimal candidates ----------
[p,b2]
[p,v]
[r1,b2]
[r1,v]
[r2,b2]
[r2,v]

**** Result assessment ****
> Global influences:
Entity <hard> put under focus because global influence
between <hard> and <air> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format:
[value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(very_low, hard, p4, []), value(low, hard, p7, [])].
Measurements |: [value(very_low, hard, p4, []), value(low, hard, p7, [])].

------- Predictions obtained about entity [hard] -----

value(very_low,hard,p5,[b2])
value(very_low,hard,p9,[v])
value(low,hard,p6,[r1])
value(low,hard,p8,[r2])
value(very_low,hard,p6,[b2,p])
value(very_low,hard,p8,[v,b2])
value(low,hard,p5,[r1,p])
value(low,hard,p9,[r2,b2])
value(very_low,hard,p7,[b2,p,r1])
value(very_low,hard,p7,[v,b2,r2])
value(low,hard,p4,[r1,p,b2])
value(low,hard,p4,[r2,b2,v])
value(very_low,hard,p8,[b2,p,r1,r2])
value(very_low,hard,p6,[v,b2,r2,r1])
value(low,hard,p9,[r1,p,b2,v])
value(very_low,hard,p5,[v,b2,r2,r1,p])

---------- New symptoms and conflict sets ----------

Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : hard = very_low vs hard = low
(resulting conflict set: [r2,b2,v])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [b2,p,r1])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [b2,r1,p])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [v,r2,b2])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r1,p,b2,v])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r1,b2,p])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2,r1])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [r2,v,b2])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,r2,r1,p])

-------- Updated minimal conflict sets --------
[p,r1,r2]
[v,b2]
[r1,p,b2]

---------- Updated minimal candidates ----------
[r2,b2]
[r1,v]
[r1,b2]
[p,v]
[p,b2]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.
The entities involved in new conflicts are not causally linked
=> End
*** End of result assessment ***
*** End of process ***

Scenario 2, ideal set of measurements

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(very_high, temp, p6, []),
value(very_high, temp, p7, [])].

------ Predictions obtained about entity [temp] ----

value(high,temp,p7,[r1])
value(very_high,temp,p5,[p])
value(high,temp,p8,[r2])
value(medium,temp,p8,[r1,r2])
value(medium,temp,p4,[p,b2])
value(very_high,temp,p9,[r1,r2,b2])
value(very_high,temp,p9,[p,b2,v])
value(medium,temp,p4,[r1,r2,b2,v])

--------- New symptoms and conflict sets ---------

Symptom at loc. p7 : temp = very_high vs temp = high
 (resulting conflict set: [r1] )
Symptom at loc. p8 : temp = high vs temp = medium
 (resulting conflict set: [r1,r2] )

------ Updated minimal conflict sets --------
[r1]

--------- Updated minimal candidates ----------
[r1]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
Entity <flow> put under focus because local influence
between <flow> and <temp> (involved in conflicts),
within component r1

Entity <air> put under focus because local influence
between <air> and <temp> (involved in conflicts),
within component r1
*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p5, []), value(medium, flow, p6, [])].

Enter the measurements for entity air, with format:
[value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(very_low, air, p6, []), value(very_low, air, p8, [])].

------ Predictions obtained about entity [flow,air] ----

value(very_low,air,p7,[r1])
value(very_low,air,p5,[p])
value(very_low,air,p9,[b2])
value(very_low,air,p7,[r2])
value(high,flow,p4,[b2])
value(high,flow,p6,[p])
value(medium,flow,p7,[r1])
value(medium,flow,p5,[p])
value(very_low,air,p8,[r1,r2])
value(very_low,air,p4,[p,b2])
value(very_low,air,p4,[b2,v])
value(very_low,air,p6,[r2,r1])
```

value(high,flow,p9,[b2,v])
value(high,flow,p7,[p,r1])
value(medium,flow,p8,[r1,r2])
value(medium,flow,p4,[p,b2])
value(very_low,air,p9,[r1,r2,b2])
value(very_low,air,p9,[p,b2,v])
value(very_low,air,p5,[r2,r1,p])
value(high,flow,p8,[p,r1,r2])
value(medium,flow,p9,[r1,r2,b2])
value(medium,flow,p9,[p,b2,v])
value(very_low,air,p4,[r1,r2,b2,v])
value(very_low,air,p4,[r2,r1,p,b2])
value(high,flow,p9,[p,r1,r2,b2])
value(medium,flow,p4,[r1,r2,b2,v])
value(very_low,air,p9,[r2,r1,p,b2,v])
value(high,flow,p4,[p,r1,r2,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p5 : flow = high vs flow = medium
(resulting conflict set: [p])
Symptom at loc. p6 : flow = medium vs flow = high
(resulting conflict set: [p])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [p,b2])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [r1,r2,b2,v])
Symptom at loc. p7 : flow = medium vs flow = high
(resulting conflict set: [p,r1])
Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [v,r1,r2,b2])
Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [p,b2,v])
Symptom at loc. p8 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [v,p,r1,r2,b2])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])

-------- Updated minimal conflict sets --------
[r1]
[p]

---------- Updated minimal candidates ----------
[r1,p]

**** Result assessment ****
> Global influences:
Entity <hard> put under focus because global influence
between <hard> and <flow> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format:
[value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(low, hard, p6, []), value(very_low, hard, p7, [])].

------- Predictions obtained about entity [hard] -----

value(low,hard,p7,[r1])
value(low,hard,p5,[p])
value(very_low,hard,p6,[r1])
value(very_low,hard,p8,[r2])
value(low,hard,p8,[r1,r2])
value(low,hard,p4,[p,b2])
value(very_low,hard,p5,[r1,p])
value(very_low,hard,p9,[r2,b2])

```
value(low,hard,p9,[r1,r2,b2])
value(low,hard,p9,[p,b2,v])
value(very_low,hard,p4,[r1,p,b2])
value(very_low,hard,p4,[r2,b2,v])
value(low,hard,p4,[r1,r2,b2,v])
value(very_low,hard,p9,[r1,p,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p6 : hard = low vs hard = very_low
 (resulting conflict set: [r1] )
Symptom at loc. p7 : hard = very_low vs hard = low
 (resulting conflict set: [r1] )
Symptom at loc. p5 : hard = low vs hard = very_low
 (resulting conflict set: [r1,p] )
Symptom at loc. p8 : hard = very_low vs hard = low
 (resulting conflict set: [r1,r2] )
Symptom at loc. p4 : hard = low vs hard = very_low
 (resulting conflict set: [r1,p,b2] )
Symptom at loc. p4 : hard = low vs hard = very_low
 (resulting conflict set: [p,r2,b2,v] )
Symptom at loc. p9 : hard = very_low vs hard = low
 (resulting conflict set: [r1,r2,b2] )
Symptom at loc. p9 : hard = very_low vs hard = low
 (resulting conflict set: [r2,p,b2,v] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r2,r1,p,b2,v] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r1,p,b2,v] )
Symptom at loc. p4 : hard = very_low vs hard = low
 (resulting conflict set: [p,r1,r2,b2,v] )
Symptom at loc. p4 : hard = very_low vs hard = low
 (resulting conflict set: [r1,r2,b2,v] )

------- Updated minimal conflict sets --------
[p]
[r1]

---------- Updated minimal candidates ----------
[p,r1]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.
The entities involved in new conflicts are not causally linked
        => End
*** End of result assessment ***
*** End of process ***
```

Scenario 2, rich set of measurements

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(medium, temp, p4, []),
value(very_high, temp, p6, []), value(very_high, temp,
p7, []), value(high, temp, p8, [])].

------- Predictions obtained about entity [temp] -----
value(very_high,temp,p5,[b2])
value(very_high,temp,p9,[v])
value(high,temp,p7,[r1])
value(very_high,temp,p5,[p])
value(high,temp,p8,[r2])
value(very_high,temp,p7,[r2])
value(very_high,temp,p6,[b2,p])
value(medium,temp,p8,[v,b2])
value(medium,temp,p8,[r1,r2])
value(medium,temp,p4,[p,b2])
value(high,temp,p7,[b2,p,r1])
```

value(high,temp,p7,[v,b2,r2])
value(very_high,temp,p9,[r1,r2,b2])
value(very_high,temp,p9,[p,b2,v])
value(medium,temp,p8,[b2,p,r1,r2])
value(very_high,temp,p6,[v,b2,r2,r1])
value(medium,temp,p4,[r1,r2,b2,v])
value(very_high,temp,p5,[v,b2,r2,r1,p])

---------- New symptoms and conflict sets ----------

Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [r1])
Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [b2,p,r1])
Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [v,b2,r2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [v,b2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p7 : temp = high vs temp = very_high
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [r2,v,b2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : temp = high vs temp = medium
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [r2,b2,p,r1])
Symptom at loc. p7 : temp = very_high vs temp = high
(resulting conflict set: [v,b2,r2])

------- Updated minimal conflict sets --------
[v,b2]
[r1]

---------- Updated minimal candidates ----------
[v,r1]
[b2,r1]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
Entity <flow> put under focus because local influence
between <flow> and <temp> (involved in conflicts),
within component r1

Entity <air> put under focus because local influence
between <air> and <temp> (involved in conflicts),
within component r1
*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format:
[value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p2, []), value(high, flow, p5, []),
value(medium, flow, p6, []), value(medium, flow, p8, [])].

Enter the measurements for entity air, with format:
[value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(very_low, air, p4, []), value(very_low, air, p5, []),
value(very_low, air, p7, []), value(very_low, air, p9, [])].

------- Predictions obtained about entity [flow,air] -----

value(very_low,air,p5,[b2])
value(very_low,air,p9,[v])
value(very_low,air,p4,[b2])
value(very_low,air,p6,[p])

value(very_low,air,p6,[r1])
value(very_low,air,p8,[r2])
value(very_low,air,p8,[b2])
value(very_low,air,p4,[v])
value(high,flow,p3,[v])
value(high,flow,p1,[s])
value(high,flow,p4,[b2])
value(high,flow,p6,[p])
value(medium,flow,p7,[r1])
value(medium,flow,p5,[p])
value(medium,flow,p9,[b2])
value(medium,flow,p7,[r2])
value(very_low,air,p6,[b2,p])
value(very_low,air,p8,[v,b2])
value(very_low,air,p9,[b2,v])
value(very_low,air,p7,[p,r1])
value(very_low,air,p5,[r1,p])
value(very_low,air,p9,[r2,b2])
value(very_low,air,p7,[b2,r2])
value(very_low,air,p5,[v,b2])
value(high,flow,p9,[b2,v])
value(high,flow,p7,[p,r1])
value(medium,flow,p8,[r1,r2])
value(medium,flow,p4,[p,b2])
value(medium,flow,p4,[b2,v])
value(medium,flow,p6,[r2,r1])
value(very_low,air,p7,[b2,p,r1])
value(very_low,air,p7,[v,b2,r2])
value(very_low,air,p8,[p,r1,r2])
value(very_low,air,p4,[r1,p,b2])
value(very_low,air,p4,[r2,b2,v])
value(very_low,air,p6,[b2,r2,r1])
value(very_low,air,p6,[v,b2,p])
value(high,flow,p8,[p,r1,r2])
value(medium,flow,p9,[r1,r2,b2])
value(medium,flow,p9,[p,b2,v])
value(medium,flow,p5,[r2,r1,p])
value(very_low,air,p8,[b2,p,r1,r2])
value(very_low,air,p6,[v,b2,r2,r1])
value(very_low,air,p9,[p,r1,r2,b2])
value(very_low,air,p9,[r1,p,b2,v])
value(very_low,air,p5,[b2,r2,r1,p])
value(very_low,air,p7,[v,b2,p,r1])
value(high,flow,p9,[p,r1,r2,b2])
value(medium,flow,p4,[r1,r2,b2,v])
value(medium,flow,p4,[r2,r1,p,b2])
value(very_low,air,p5,[v,b2,r2,r1,p])
value(very_low,air,p4,[p,r1,r2,b2,v])
value(very_low,air,p8,[v,b2,p,r1,r2])
value(high,flow,p4,[p,r1,r2,b2,v])
value(medium,flow,p9,[r2,r1,p,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p5 : flow = high vs flow = medium
(resulting conflict set: [p])
Symptom at loc. p5 : flow = high vs flow = medium
(resulting conflict set: [r2,r1,p])
Symptom at loc. p6 : flow = medium vs flow = high
(resulting conflict set: [p])
Symptom at loc. p8 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [p,b2])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [b2,v])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [r1,r2,b2,v])
Symptom at loc. p4 : flow = high vs flow = medium
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p6 : flow = high vs flow = medium
(resulting conflict set: [p,r2,r1])
Symptom at loc. p7 : flow = medium vs flow = high
(resulting conflict set: [p,r1])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [b2,v])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2])
Symptom at loc. p7 : flow = medium vs flow = high
(resulting conflict set: [r2,p,r1])
Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [v,r1,r2,b2])

Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [p,b2,v])
Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p8 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2])
Symptom at loc. p9 : flow = medium vs flow = high
(resulting conflict set: [v,p,r1,r2,b2])
Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])
Symptom at loc. p4 : flow = medium vs flow = high
(resulting conflict set: [p,r1,r2,b2,v])

------ Updated minimal conflict sets --------
[r1]
[b2,v]
[p]

---------- Updated minimal candidates ----------
[r1,b2,p]
[r1,v,p]

**** Result assessment ****
> Global influences:
Entity <hard> put under focus because global influence between <hard> and <flow> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go]
followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format:
[value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(low, hard, p4, []), value(low, hard, p6, []), value(very_low, hard, p7, []), value(very_low, hard, p8, [])].

-------- Predictions obtained about entity [hard] -----

value(low,hard,p5,[b2])
value(low,hard,p9,[v])
value(low,hard,p7,[r1])
value(low,hard,p5,[p])
value(very_low,hard,p6,[r1])
value(very_low,hard,p8,[r2])
value(very_low,hard,p9,[b2])
value(very_low,hard,p7,[r2])
value(low,hard,p6,[b2,p])
value(low,hard,p8,[v,b2])
value(low,hard,p8,[r1,r2])
value(low,hard,p4,[p,b2])
value(very_low,hard,p5,[r1,p])
value(very_low,hard,p9,[r2,b2])
value(very_low,hard,p4,[b2,v])
value(very_low,hard,p6,[r2,r1])
value(low,hard,p7,[b2,p,r1])
value(low,hard,p7,[v,b2,r2])
value(low,hard,p9,[r1,r2,b2])
value(low,hard,p9,[p,b2,v])
value(very_low,hard,p4,[r1,p,b2])
value(very_low,hard,p4,[r2,b2,v])
value(very_low,hard,p5,[r2,r1,p])
value(low,hard,p8,[b2,p,r1,r2])
value(low,hard,p6,[v,b2,r2,r1])
value(low,hard,p4,[r1,r2,b2,v])
value(very_low,hard,p9,[r1,p,b2,v])

value(very_low,hard,p4,[r2,r1,p,b2])
value(low,hard,p5,[v,b2,r2,r1,p])
value(very_low,hard,p9,[r2,r1,p,b2,v])

---------- New symptoms and conflict sets ---------

Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [b2,v])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r2,b2,v])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r1])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [r1])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [b2,p,r1])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [v,b2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [b2,r1,p])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1,p])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [v,b2])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [v,r2,b2])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [r1,p,b2,v])
Symptom at loc. p9 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,p,b2,v])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r1,r2])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [r1,p])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,p])
Symptom at loc. p6 : hard = very_low vs hard = low
(resulting conflict set: [r1,b2,p])
Symptom at loc. p6 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [r2,v,b2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [r1,r2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [p,b2,v])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [r2,b2,p,r1])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [b2,p,r2,r1])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [p,b2,v])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [p,r2,b2,v])
Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [r2,r1,p,b2])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1,p])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r1,r2,b2])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [r2,p,b2,v])

```
Symptom at loc. p4 : hard = very_low vs hard = low
 (resulting conflict set: [r1,r2,b2,v] )
Symptom at loc. p6 : hard = very_low vs hard = low
 (resulting conflict set: [v,b2,r2,r1] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r2,r1,p,b2,v] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r2,r1,p,b2,v] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r1,p,b2,v] )
Symptom at loc. p9 : hard = low vs hard = very_low
 (resulting conflict set: [r2,r1,p,b2,v] )
Symptom at loc. p4 : hard = very_low vs hard = low
 (resulting conflict set: [p,r1,r2,b2,v] )
Symptom at loc. p4 : hard = very_low vs hard = low
 (resulting conflict set: [r1,r2,b2,v] )
Symptom at loc. p5 : hard = very_low vs hard = low
 (resulting conflict set: [v,b2,r2,r1,p] )
Symptom at loc. p4 : hard = low vs hard = very_low
 (resulting conflict set: [v,r2,r1,p,b2] )

-------- Updated minimal conflict sets ---------
[p]
[r1]
[b2,v]

---------- Updated minimal candidates ----------
[p,r1,v]
[p,r1,b2]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.
The entities involved in new conflicts are not causally linked
        => End
*** End of result assessment ***
*** End of process ***
```

Scenario 2, poor set of measurements

```
| ?- candidates([temp]).

Enter the measurements for entity temp, with format:
[value(qualitat. value, temp, location, []), value(...), ...].

Measurements |: [value(very_high, temp, p6, []),
value(high, temp, p8, [])].

-------- Predictions obtained about entity [temp] -----

value(high,temp,p7,[r1])
value(very_high,temp,p5,[p])
value(very_high,temp,p7,[r2])
value(medium,temp,p8,[r1,r2])
value(medium,temp,p4,[p,b2])
value(very_high,temp,p9,[r1,r2,b2])
value(very_high,temp,p9,[p,b2,v])
value(medium,temp,p4,[r1,r2,b2,v])

---------- New symptoms and conflict sets ----------

Symptom at loc. p8 : temp = high vs temp = medium
 (resulting conflict set: [r1,r2] )
Symptom at loc. p7 : temp = high vs temp = very_high
 (resulting conflict set: [r1,r2] )

-------- Updated minimal conflict sets ---------
[r1,r2]

---------- Updated minimal candidates ----------
[r2]
[r1]

**** Result assessment ****
```

> Global influences:

No global influence of interest.

> Local influences:

Entity <flow> put under focus because local influence between <flow> and <temp> (involved in conflicts), within component r1

Entity <air> put under focus because local influence between <air> and <temp> (involved in conflicts), within component r1

*** End of result assessment and start of dynamic revision***

The new entities under focus are [flow,air]

*** End of dynamic revision ***

To terminate type [end], to go further type [go] followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity flow, with format: [value(qualitat. value, flow, location, []), value(...), ...].

Measurements |: [value(high, flow, p4, []), value(medium, flow, p7, [])].

Enter the measurements for entity air, with format: [value(qualitat. value, air, location, []), value(...), ...].

Measurements |: [value(very_low, air, p4, []), value(very_low, air, p8, [])].

------- Predictions obtained about entity [flow,air] -----

value(very_low,air,p5,[b2])
value(very_low,air,p9,[v])
value(very_low,air,p9,[b2])
value(very_low,air,p7,[r2])
value(high,flow,p5,[b2])
value(high,flow,p9,[v])
value(medium,flow,p6,[r1])
value(medium,flow,p8,[r2])
value(very_low,air,p6,[b2,p])
value(very_low,air,p8,[v,b2])
value(very_low,air,p4,[b2,v])
value(very_low,air,p6,[r2,r1])
value(high,flow,p6,[b2,p])
value(high,flow,p8,[v,b2])
value(medium,flow,p5,[r1,p])
value(medium,flow,p9,[r2,b2])
value(very_low,air,p7,[b2,p,r1])
value(very_low,air,p7,[v,b2,r2])
value(very_low,air,p5,[r2,r1,p])
value(high,flow,p7,[b2,p,r1])
value(high,flow,p7,[v,b2,r2])
value(medium,flow,p4,[r1,p,b2])
value(medium,flow,p4,[r2,b2,v])
value(very_low,air,p8,[b2,p,r1,r2])
value(very_low,air,p6,[v,b2,r2,r1])
value(very_low,air,p4,[r2,r1,p,b2])
value(high,flow,p8,[b2,p,r1,r2])
value(high,flow,p6,[v,b2,r2,r1])
value(medium,flow,p9,[r1,p,b2,v])
value(very_low,air,p5,[v,b2,r2,r1,p])
value(very_low,air,p9,[r2,r1,p,b2,v])
value(high,flow,p5,[v,b2,r2,r1,p])

---------- New symptoms and conflict sets ----------

Symptom at loc. p4 : flow = high vs flow = medium (resulting conflict set: [r1,p,b2])
Symptom at loc. p4 : flow = high vs flow = medium (resulting conflict set: [r2,b2,v])
Symptom at loc. p7 : flow = medium vs flow = high (resulting conflict set: [b2,p,r1])
Symptom at loc. p7 : flow = medium vs flow = high (resulting conflict set: [v,b2,r2])
Symptom at loc. p5 : flow = high vs flow = medium (resulting conflict set: [b2,r1,p])
Symptom at loc. p9 : flow = high vs flow = medium (resulting conflict set: [v,r2,b2])

Symptom at loc. p9 : flow = high vs flow = medium
(resulting conflict set: [r1,p,b2,v])
Symptom at loc. p6 : flow = medium vs flow = high
(resulting conflict set: [r1,b2,p])
Symptom at loc. p6 : flow = medium vs flow = high
(resulting conflict set: [v,b2,r2,r1])
Symptom at loc. p8 : flow = medium vs flow = high
(resulting conflict set: [r2,v,b2])
Symptom at loc. p8 : flow = medium vs flow = high
(resulting conflict set: [b2,p,r1,r2])
Symptom at loc. p5 : flow = medium vs flow = high
(resulting conflict set: [v,b2,r2,r1,p])

------- Updated minimal conflict sets --------
[r1,r2]
[r2,b2,v]
[r1,p,b2]

---------- Updated minimal candidates ----------
[r2,b2]
[r2,p]
[r2,r1]
[r1,b2]
[r1,v]

**** Result assessment ****
> Global influences:
Entity <hard> put under focus because global influence between <hard> and <flow> (involved in conflicts)
> Local influences:
No local influence of interest.
*** End of result assessment and start of dynamic revision***

The new entities under focus are [hard]
*** End of dynamic revision ***

To terminate type [end], to go further type [go] followed by a dot and press the [enter] key.|: go.

*** Start of further diagnostic reasoning ***

Enter the measurements for entity hard, with format:
[value(qualitat. value, hard, location, []), value(...), ...].

Measurements |: [value(low, hard, p4, []), value(very_low, hard, p9, [])].

------- Predictions obtained about entity [hard] -----

value(low,hard,p5,[b2])
value(low,hard,p9,[v])
value(very_low,hard,p8,[b2])
value(very_low,hard,p4,[v])
value(low,hard,p6,[b2,p])
value(low,hard,p8,[v,b2])
value(very_low,hard,p7,[b2,r2])
value(very_low,hard,p5,[v,b2])
value(low,hard,p7,[b2,p,r1])
value(low,hard,p7,[v,b2,r2])
value(very_low,hard,p6,[b2,r2,r1])
value(very_low,hard,p6,[v,b2,p])
value(low,hard,p8,[b2,p,r1,r2])
value(low,hard,p6,[v,b2,r2,r1])
value(very_low,hard,p5,[b2,r2,r1,p])
value(very_low,hard,p7,[v,b2,p,r1])
value(low,hard,p5,[v,b2,r2,r1,p])
value(very_low,hard,p8,[v,b2,p,r1,r2])

---------- New symptoms and conflict sets ----------

Symptom at loc. p4 : hard = low vs hard = very_low
(resulting conflict set: [v])
Symptom at loc. p9 : hard = very_low vs hard = low
(resulting conflict set: [v])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [v,b2])
Symptom at loc. p5 : hard = low vs hard = very_low
(resulting conflict set: [b2,r2,r1,p])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [v,b2])
Symptom at loc. p8 : hard = very_low vs hard = low
(resulting conflict set: [b2,p,r1,r2])

Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [p,b2,r2,r1])
Symptom at loc. p6 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p,r1,r2])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [r2,b2,p,r1])
Symptom at loc. p7 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1,p])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p,r1])
Symptom at loc. p7 : hard = low vs hard = very_low
(resulting conflict set: [r2,v,b2,p,r1])
Symptom at loc. p6 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1])
Symptom at loc. p6 : hard = very_low vs hard = low
(resulting conflict set: [p,v,b2,r2,r1])
Symptom at loc. p8 : hard = low vs hard = very_low
(resulting conflict set: [v,b2,p,r1,r2])
Symptom at loc. p5 : hard = very_low vs hard = low
(resulting conflict set: [v,b2,r2,r1,p])

------- Updated minimal conflict sets --------
[r1,r2]
[r1,p,b2]
[v]

--------- Updated minimal candidates ----------
[r1,v]
[r2,p,v]
[r2,b2,v]

**** Result assessment ****
> Global influences:
No global influence of interest.
> Local influences:
No local influence of interest.

The entities involved in new conflicts are not causally linked
=> End
*** End of result assessment ***
*** End of process ***